Texture

The most fundamental question is: How can we “measure” texture, i.e., how can we quantitatively distinguish between different textures?

Of course it is not enough to look at the intensity of individual pixels.

Since the repetitive local arrangement of intensity determines the texture, we have to analyze neighborhoods of pixels to measure texture properties.

Frequency Descriptors

One possible approach is to perform local Fourier transforms of the image.

Then we can derive information on
- the contribution of different spatial frequencies and
- the dominant orientation(s) in the local texture.

For both kinds of information, only the power (magnitude) spectrum needs to be analyzed.

Frequency Descriptors

Prior to the Fourier transform, multiply the image with a Gaussian function to avoid horizontal and vertical “phantom” lines.

In the power spectrum, use ring filters of different radii to extract the frequency band contributions.

Also in the power spectrum, apply wedge filters at different angles to obtain the information on dominant orientation of edges in the texture.

Frequency Descriptors

The resulting frequency and orientation data can be normalized, for example, so that the sum across frequency or orientation bands is 1.

This effectively turns them into histograms that are less affected by monotonic gray-level changes caused by shading etc.

However, it is recommended to combine frequency-based approaches with space-based approaches.
Co-Occurrence Matrices

A simple and popular method for this kind of analysis is the computation of **gray-level co-occurrence matrices**.

To compute such a matrix, we first separate the intensity in the image into a small number of different **levels**.

For example, by dividing the usual brightness values ranging from 0 to 255 by 64, we create the levels 0, 1, 2, and 3.

Then we choose a **displacement vector** \( \mathbf{d} = [d_i, d_j] \).

The gray-level co-occurrence matrix \( P(a, b) \) is then obtained by counting all pairs of pixels separated by \( \mathbf{d} \) having gray levels \( a \) and \( b \).

Afterwards, to **normalize the matrix**, we determine the sum across all entries and divide each entry by this sum.

This co-occurrence matrix contains important information about the texture in the examined area of the image.

**Example (2 gray levels):**

| 0 1 0 0 1 0 |
| 1 1 0 1 1 0 |
| 0 1 0 0 1 0 |
| 1 1 0 1 1 0 |
| 0 1 0 0 1 0 |
| 1 1 0 1 1 0 |

local texture patch

**displacement vector**

co-occurrence matrix

\( d = (1, 1) \)

\( \frac{1}{25} \times \begin{pmatrix} 2 & 9 & 0 \\ 10 & 4 & 1 \end{pmatrix} \)

It is often a good idea to use **more than one** displacement vector, resulting in multiple co-occurrence matrices.

The more similar the matrices of two textures are, the more similar are usually the textures themselves.

This means that the difference between corresponding elements of these matrices can be taken as a **similarity measure** for textures.

Based on such measures we can use texture information to **enhance** the detection of **regions** and **contours** in images.

For a given co-occurrence matrix \( P(a, b) \), we can compute the following six important characteristics:

- **Energy**
  \[ \sum_{a,b} P^2(a, b) \]

- **Entropy**
  \[ \sum_{a,b} P(a, b) \log_2 P(a, b) \]

- **Maximum probability**
  \[ y = \max_{a,b} P(a, b) \]

- **Contrast**
  \[ \sum_{a,b} |a - b|^\kappa \cdot P^\lambda (a, b), \text{ usually } \kappa = 2, \lambda = 1 \]

**Inverse difference moment**

\[ \sum_{a,b} P^4(a, b) \]

**Correlation**

\[ \frac{\sum_{a,b} (ab)P(a, b) - \mu_x \mu_y}{\sigma_x \sigma_y} \]

\[ \mu_x = \sum_{a} a P(a, b) \quad \sigma_x = \sum_{a} (a - \mu_x)^2 \sum_{a} P(a, b) \]

\[ \mu_y = \sum_{b} b P(a, b) \quad \sigma_y = \sum_{b} (b - \mu_y)^2 \sum_{b} P(a, b) \]
Co-Occurrence Matrices

You should compute these six characteristics for multiple displacement vectors, including different directions.

<table>
<thead>
<tr>
<th>Starting point</th>
<th>End point</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 0.2 0.3</td>
<td></td>
</tr>
<tr>
<td>1.0 1.0 1.0</td>
<td>2.0 2.0 2.2</td>
</tr>
<tr>
<td>3.0 3.0 3.0</td>
<td>5.0 5.0 5.0</td>
</tr>
</tbody>
</table>

The maximum length of your displacement vectors depends on the size of the texture elements.

Law’s Texture Energy Measures

Law’s measures use a set of convolution filters to assess gray level, edges, spots, ripples, and waves in textures.

This method starts with three basic filters:

- averaging: $L_3 = (1, 2, 1)$
- first derivative (edges): $E_3 = (-1, 0, 1)$
- second derivative (curvature): $S_3 = (-1, 2, -1)$

Now we can multiply any two of these vectors, using the first one as a column vector and the second one as a row vector, resulting in $5 \times 5$ Law’s masks.

For example:

$$L_5^T \times S_5 = \begin{bmatrix} -1 & 0 & 2 & 0 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 0 & 12 & 0 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & 0 & 2 & 0 & -1 \end{bmatrix}$$

Law’s Texture Energy Measures

Convolving these filters with themselves and each other results in five new filters:

- $L_5 = (1, 4, 6, 4, 1)$
- $E_5 = (-1, -2, 0, 2, 1)$
- $S_5 = (-1, 0, 2, 0, -1)$
- $R_5 = (1, -4, 6, -4, 1)$
- $W_5 = (-1, 2, 0, -2, 1)$

Now you can apply the resulting 25 convolution filters to a given image.

The 25 resulting values at each position in the image are useful descriptors of the local texture.

Law’s texture energy measures are easy to apply, efficient, and give good results for most texture types. However, co-occurrence matrices are more flexible; for example, they can be scaled to account for coarse-grained textures.
Local Binary Patterns

Clearly, \textbf{local intensity gradients} are important in describing textures. However, the \textbf{orientation} of these gradients may be more important than their \textbf{magnitude} (steepness). After all, changing the contrast of a picture does not fundamentally change the texture, while it modifies the magnitude of all gradients.

The technique of \textbf{local binary patterns} uses histograms to represent the relative frequencies of different gradients.

Here, the gradient at a given pixel \( p \) is computed by comparing its intensity with that of \textbf{a number of neighboring pixels}. Those could be immediate neighbors or be located at larger distances. Such neighborhoods are characterized by the number of neighbors and their distance from \( p \) ("radius"). For example, a popular choice is an \((8, 2)\) neighborhood, which uses 8 neighbors at a radius of 2 pixels.

When characterizing gradients, we are typically only interested in \textbf{uniform} binary patterns. These are the patterns that contain at most two transitions from 0 to 1 or vice versa when we read them along the full circle. The example pattern 10000011 is \textbf{uniform}, because it has only two transitions, which are between positions 1 and 2 and between positions 6 and 7 in the binary string. The pattern 11001100, on the other hand, has four transitions and is thus \textbf{not uniform}.

The total number of uniform patterns is \textbf{58}. We build a histogram with \textbf{59 bins}, with each of the first 58 bins indicating how often each of the uniform patterns was found within the image area \( P \) that we want to analyze. The 59th bin shows how many \textbf{non-uniform} patterns were encountered. We \textbf{normalize} the histograms by dividing each of its 59 entries by \(| P |\). The resulting 59-dimensional vector is the LBP descriptor of the texture patch.
Classification Performance

Texture Segmentation Benchmarks

Benchmark image for texture segmentation – an ideal segmentation algorithm would divide this image into five segments.

For example, a texture-descriptor based variant of split-and-merge may be able to achieve good results.