Introduction to Computer Vision: Segmentation by Clustering

Navasardyan Shant

October 31, 2019

Overview

Overview

What is Texture?

Texture

Texture is something uncertain, but intuitively, we, humans, can distinguish one texture from another.

What is Texture?

Texture

Texture is something uncertain, but intuitively, we, humans, can distinguish one texture from another.

Figure: Is this a texture?

What is Texture?

Figure: And this?

What is Texture?

Figure: What about this?

What is Texture?

Texton

So textures can be described as a large amount of small objects (e.g. grass, hair, wood, etc.).

Sometimes you can see in a texture some repeating pattern. The elements of this pattern are called **textons**.

What is Texture?

Texton

So textures can be described as a large amount of small objects (e.g. grass, hair, wood, etc.).
Sometimes you can see in a texture some repeating pattern. The elements of this pattern are called **textons**.

Texture

So, roughly speaking, *texture* is a domain on the image, which consists of *elements* repeating in *some way*.

Overview

Filters

As for representing a texture in an image for each pixel (which belongs to that texture region) we need to use neighborhood of that pixel, local features of the image can be quite useful.

Filters

As for representing a texture in an image for each pixel (which belongs to that texture region) we need to use neighborhood of that pixel, local features of the image can be quite useful.

We do assumption that textons are made of some known small elements (spots, bars, etc.). This elements we detect by using filters of various scales (the elements can be in various scales).

Filters

As for representing a texture in an image for each pixel (which belongs to that texture region) we need to use neighborhood of that pixel, local features of the image can be quite useful.

We do assumption that textons are made of some known small elements (spots, bars, etc.). This elements we detect by using filters of various scales (the elements can be in various scales).

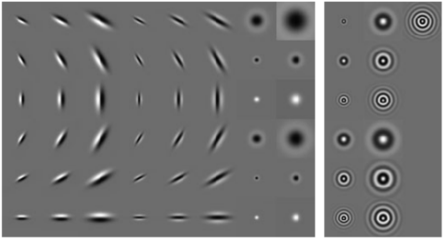


Figure: Filters for texture representation in various scales and orientations

Filters

Note

After filtering with above-mentioned filters, one can obtain feature-vectors for each pixel with some processing. For example, the absolute value or ReLU function can be used.

Filters

Note

After filtering with above-mentioned filters, one can obtain feature-vectors for each pixel with some processing. For example, the absolute value or ReLU function can be used.

Applications

Local texture representation can be used in *texture classification, texture segmentation, image segmentation with clustering, etc.*

Overview

Pooled Texture Representation

Vector Quantization

In some cases a description of a larger domain is also useful. After local texture representation, it can be a useful information to know "how many" textons are in that domain. Since we can not directly compute the number of these textons, we apply a technique called **vector quantization**.

Pooled Texture Representation

Vector Quantization

In some cases a description of a larger domain is also useful. After local texture representation, it can be a useful information to know "how many" textons are in that domain. Since we can not directly compute the number of these textons, we apply a technique called **vector quantization**.

To obtain the estimation of the "number" of particular textons in the domain, we first compute some local features (it can be, for example, the local texture representation) of patches from some *training set*.

Pooled Texture Representation

Vector Quantization

In some cases a description of a larger domain is also useful. After local texture representation, it can be a useful information to know "how many" textons are in that domain. Since we can not directly compute the number of these textons, we apply a technique called **vector quantization**.

To obtain the estimation of the "number" of particular textons in the domain, we first compute some local features (it can be, for example, the local texture representation) of patches from some *training set*.
After, with predefined k, we do k-means clustering on these features and find the cluster centers $c_1, \ldots c_k$ (which are pixel-level local features).

Pooled Texture Representation

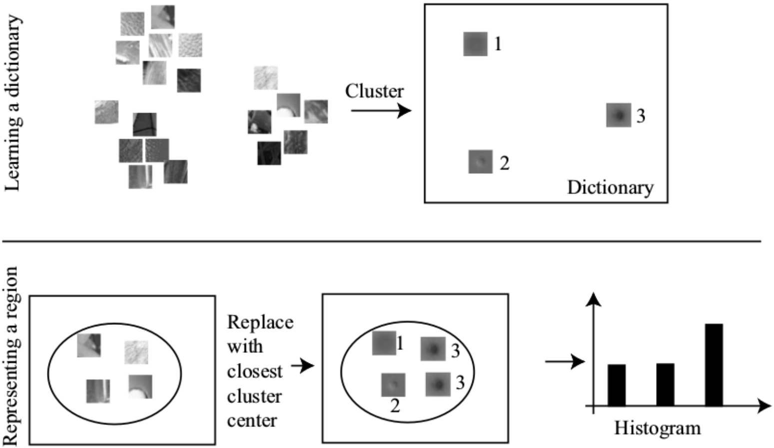
Vector Quantization

In some cases a description of a larger domain is also useful. After local texture representation, it can be a useful information to know "how many" textons are in that domain. Since we can not directly compute the number of these textons, we apply a technique called **vector quantization**.

To obtain the estimation of the "number" of particular textons in the domain, we first compute some local features (it can be, for example, the local texture representation) of patches from some *training set*.
After, with predefined k, we do k-means clustering on these features and find the cluster centers $c_1, \dots c_k$ (which are pixel-level local features).
Then for each pixel in the domain, we find its feature-vector, then compute the closest cluster centers for these feature vectors. After finding the corresponding c_i for each pixel in the domain, we compute the frequencies (histogram) of these centers.

Illustration

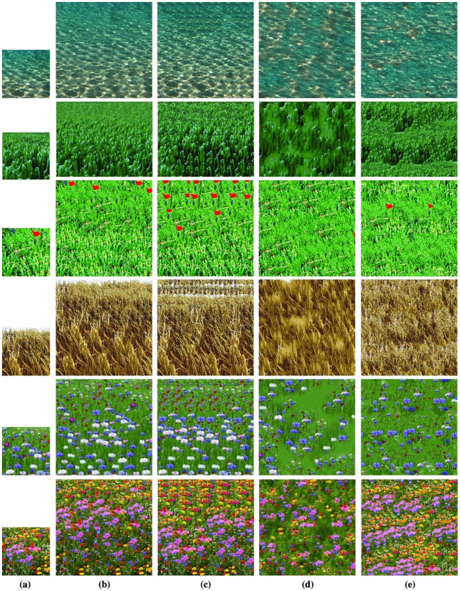
In the following image you can see an illustration of the process described above:



Overview

Texture Synthesis

Sometimes we have a small sample of a texture and we want to generate the whole texture.



Texture Synthesis

The Problem

Let we have a texture sample $I_{smp} \in \mathbb{R}^{H \times W \times C}$, and suppose it was sampled from a real *"infinite"* texture I_{real}.
The problem is to describe a method to predict the value of I_{real} at every point $(i, j) \in \mathbb{Z}^2$.
Essentially, the problem is for any integers $H, W > 0$ obtain an image $I_{real} \in \mathbb{R}^{H \times W \times C}$ with the same texture as the sample I_{smp}.

Texture Synthesis

The Problem

Let we have a texture sample $I_{smp} \in \mathbb{R}^{H \times W \times C}$, and suppose it was sampled from a real *"infinite"* texture I_{real}.
The problem is to describe a method to predict the value of I_{real} at every point $(i, j) \in \mathbb{Z}^2$.
Essentially, the problem is for any integers $H, W > 0$ obtain an image $I_{real} \in \mathbb{R}^{H \times W \times C}$ with the same texture as the sample I_{smp}.

The following algorithm originally is from the paper *"Texture Synthesis by Non-Parametric Sampling", A. A. Efros, T. K. Leung, Proc. IEEE ICCV, 1999.*

Synthesizing One Pixel

First let's assume we have the image I_{real} except of one pixel $p \in \mathbb{Z}^2$ and we want to predict the value of I_{real} in p. Let $\omega(p) \in \mathbb{R}^{k \times k \times C}$ be a square image patch (of size $k \times k$) centered at p.

Synthesizing One Pixel

First let's assume we have the image I_{real} except of one pixel $p \in \mathbb{Z}^2$ and we want to predict the value of I_{real} in p. Let $\omega(p) \in \mathbb{R}^{k \times k \times C}$ be a square image patch (of size $k \times k$) centered at p. For all patches $\omega \in \mathbb{R}^{k \times k \times C}$ which do not contain the pixel indexed by p, we define some **distance** $d(\omega, \omega(p))$. For example, one can take $d(a, b) = \|a - b\|^2$.

Synthesizing One Pixel

First let's assume we have the image I_{real} except of one pixel $p \in \mathbb{Z}^2$ and we want to predict the value of I_{real} in p. Let $\omega(p) \in \mathbb{R}^{k \times k \times C}$ be a square image patch (of size $k \times k$) centered at p. For all patches $\omega \in \mathbb{R}^{k \times k \times C}$ which do not contain the pixel indexed by p, we define some **distance** $d(\omega, \omega(p))$. For example, one can take $d(a, b) = \|a - b\|^2$. Then, among all patches ω, which do not contain the pixel indexed by p, we find the best match in terms of the distance d:

$$\omega_{best} = argmin_{\omega} d(\omega, \omega(p)).$$

Synthesizing One Pixel

First let's assume we have the image I_{real} except of one pixel $p \in \mathbb{Z}^2$ and we want to predict the value of I_{real} in p. Let $\omega(p) \in \mathbb{R}^{k \times k \times C}$ be a square image patch (of size $k \times k$) centered at p. For all patches $\omega \in \mathbb{R}^{k \times k \times C}$ which do not contain the pixel indexed by p, we define some **distance** $d(\omega, \omega(p))$. For example, one can take $d(a, b) = \|a - b\|^2$. Then, among all patches ω, which do not contain the pixel indexed by p, we find the best match in terms of the distance d:

$$\omega_{best} = argmin_{\omega} d(\omega, \omega(p)).$$

Then for a parameter ε we consider the set

$$\Omega(p) = \{\omega \ : \ d(\omega, \omega(p)) \leq (1 + \varepsilon) d(\omega(p), \omega_{best})\},$$

uniformly sample a patch, and fill the value of p with the value of the center of the chosen patch.

Synthesizing Texture

Now for any point $p \in \mathbb{Z}^2$ to be synthesized only *some* of the pixel values in $\omega(p)$ are known. Let the set of known neighbors of p be K. Then for any patch ω, for which all pixels in K are known, we consider the distance

$$d'(\omega, \omega(p)) = \frac{1}{|K|} \sum_{(i,j) \in K} dist(\omega_{i,j}, \omega(p)_{i,j}),$$

where *dist* is some distance between two pixels (for example s.s.d. - sum of squared differences).

Synthesizing Texture

Now for any point $p \in \mathbb{Z}^2$ to be synthesized only *some* of the pixel values in $\omega(p)$ are known. Let the set of known neighbors of p be K. Then for any patch ω, for which all pixels in K are known, we consider the distance

$$d'(\omega, \omega(p)) = \frac{1}{|K|} \sum_{(i,j) \in K} dist(\omega_{i,j}, \omega(p)_{i,j}),$$

where *dist* is some distance between two pixels (for example s.s.d. - sum of squared differences). Then, by the same way, we construct the set

$$\Omega(p) = \{\omega \ : \ d'(\omega, \omega(p)) \leq (1+\varepsilon) d'(\omega(p), \omega_{best})\},$$

uniformly sample from it a patch, and fill the value of p with the value of the center of the chosen patch.

Synthesizing Texture

Now for any point $p \in \mathbb{Z}^2$ to be synthesized only *some* of the pixel values in $\omega(p)$ are known. Let the set of known neighbors of p be K. Then for any patch ω, for which all pixels in K are known, we consider the distance

$$d'(\omega, \omega(p)) = \frac{1}{|K|} \sum_{(i,j) \in K} dist(\omega_{i,j}, \omega(p)_{i,j}),$$

where *dist* is some distance between two pixels (for example s.s.d. - sum of squared differences). Then, by the same way, we construct the set

$$\Omega(p) = \{\omega \ : \ d'(\omega, \omega(p)) \le (1+\varepsilon) d'(\omega(p), \omega_{best})\},$$

uniformly sample from it a patch, and fill the value of p with the value of the center of the chosen patch.

Note

Initially, we can refer the image I_{smp} as already filled pixels and try to extend it to the size $H \times W$.

Introduction to Computer Vision: Segmentation by Clustering

Navasardyan Shant

October 31, 2019