matchTemplate

Compares a template against overlapped image regions.

**C++:**void **matchTemplate**(InputArray **image**, InputArray **templ**, OutputArray **result**, int **method**)

**Python:**cv2.**matchTemplate**(image, templ, method[, result]) → result

**C:**void **cvMatchTemplate**(const CvArr\* **image**, const CvArr\* **templ**, CvArr\* **result**, int **method**)

**Python:**cv.**MatchTemplate**(image, templ, result, method) → None

|  |  |
| --- | --- |
| **Parameters:** | * **image** – Image where the search is running. It must be 8-bit or 32-bit floating-point. * **templ** – Searched template. It must be not greater than the source image and have the same data type. * **result** – Map of comparison results. It must be single-channel 32-bit floating-point. If image is W \times H and templ is w \times h , then result is (W-w+1) \times (H-h+1) . * **method** – Parameter specifying the comparison method (see below). |

The function slides through image , compares the overlapped patches of size w \times h against templ using the specified method and stores the comparison results in result . Here are the formulae for the available comparison methods ( I denotes image, T template, R result ). The summation is done over template and/or the image patch: x' = 0...w-1, y' = 0...h-1

* method=CV\_TM\_SQDIFF

R(x,y)= \sum _{x',y'} (T(x',y')-I(x+x',y+y'))^2

* method=CV\_TM\_SQDIFF\_NORMED

R(x,y)= \frac{\sum_{x',y'} (T(x',y')-I(x+x',y+y'))^2}{\sqrt{\sum_{x',y'}T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}

* method=CV\_TM\_CCORR

R(x,y)= \sum _{x',y'} (T(x',y')  \cdot I(x+x',y+y'))

* method=CV\_TM\_CCORR\_NORMED

R(x,y)= \frac{\sum_{x',y'} (T(x',y') \cdot I(x+x',y+y'))}{\sqrt{\sum_{x',y'}T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}

* method=CV\_TM\_CCOEFF

R(x,y)= \sum _{x',y'} (T'(x',y')  \cdot I'(x+x',y+y'))

where

\begin{array}{l} T'(x',y')=T(x',y') - 1/(w  \cdot h)  \cdot \sum _{x'',y''} T(x'',y'') \\ I'(x+x',y+y')=I(x+x',y+y') - 1/(w  \cdot h)  \cdot \sum _{x'',y''} I(x+x'',y+y'') \end{array}

* method=CV\_TM\_CCOEFF\_NORMED

R(x,y)= \frac{ \sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y')) }{ \sqrt{\sum_{x',y'}T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2} }

After the function finishes the comparison, the best matches can be found as global minimums (when CV\_TM\_SQDIFF was used) or maximums (when CV\_TM\_CCORR or CV\_TM\_CCOEFF was used) using the **[minMaxLoc()](https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html" \l "void%20minMaxLoc(InputArray%20src,%20double*%20minVal,%20double*%20maxVal,%20Point*%20minLoc,%20Point*%20maxLoc,%20InputArray%20mask)" \o "void minMaxLoc(InputArray src, double* minVal, double* maxVal, Point* minLoc, Point* maxLoc, InputArray mask))** function. In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, which is easier to analyze.