Scalable Local Feature Extraction with Orientation Maps and GPU Computing

National Institute of Advanced Industrial Science and Technology
AIST

Naoyuki Ichimura
AIST, Japan
nic@ni.aist.go.jp

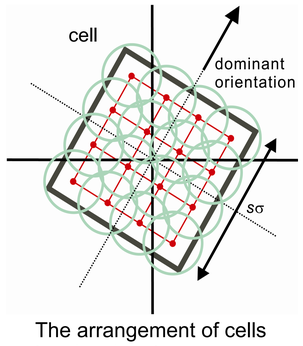
Local Invariant Features

The purpose of this research is to improve the scalability of feature extraction in terms of the number of features.

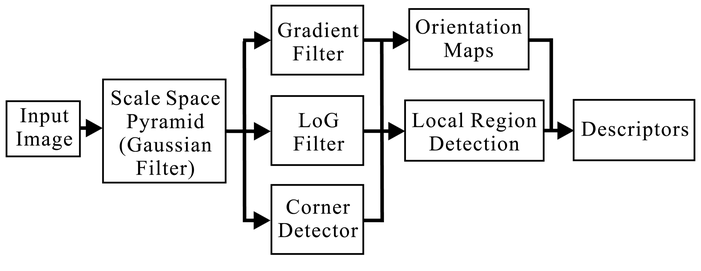
An example of detecting local regions. Dense detection such as this is desirable in improving performance in image matching and object recognition using local invariant features.

[Calculating Local Descriptors]

1. The histograms of gradient orientations are computed in cells arranged in a local region. The voting values for the histograms are the gradient magnitudes of the pixels in the cells.
2. A local descriptor is obtained by concatenating the histograms of the cells.



The arrangement of cells in a local region.

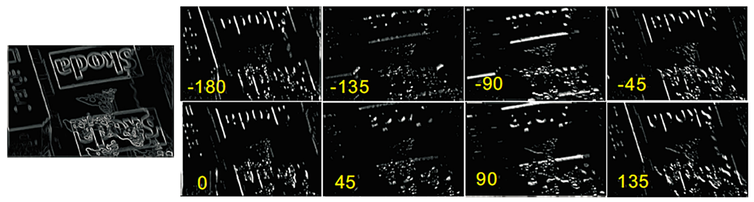


The flowchart of the implemented feature extraction algorithm. Both feature points and edges are used for local region detection. Local descriptors consisting of the histograms of gradient orientations are calculated using orientation maps.

Orientation Maps

The voting values for the histograms of gradient orientations can be computed by applying multiscale Gaussian filters to 2D arrays corresponding to the bins of the histograms of gradient orientations, i.e., *orientation maps*.

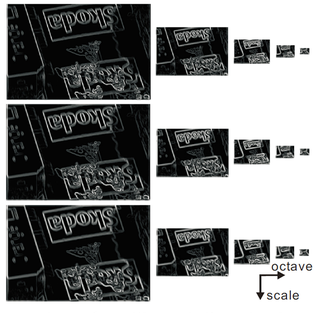
This allows us to calculate the descriptors merely by looking up the locations of convolved orientation maps corresponding to the centers of cells. No exhaustive access to the pixels in cells is required thanks to orientation maps.



Examples of orientation maps corresponding to the eight quantized gradient orientations. Each of the maps contains gradient magnitudes for a quantized orientation obtained from the edge image on the left.

Using orientation maps reduces the number of redundant computations in calculating local descriptors due to the overlaps among local regions.

However, many convolution operations need to be performed on large numbers of multiresolutional orientation maps.



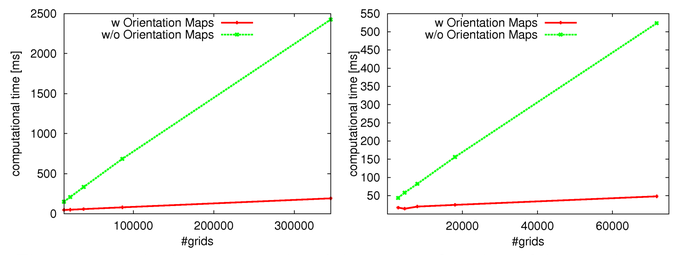
Examples of multiresolutional edge images. Multiresolutional orientation maps are obtained by computing the orientation maps for each edge image.

We can take advantage of GPU computing in calculating the local descriptors using orientation maps:

- Computing the voting values by multiscale Gaussian filters can be efficiently implemented by utilizing on-chip memory of a GPU.

Performance Results

The test platform uses a 3.33GHz Intel Xeon X5680 CPU, an NVIDIA GTX580 graphics card, CUDA4.0 and the Linux. Using orientation maps and the GPU enable us to improve the scalability of feature extraction in terms of the number of features as well as the efficiency of the computation compared to the CPU (25x-40x).



The changes in the computational times as a function of the number of grids on which features are extracted. Note the clear improvement of the scalability of feature extraction using the orientation maps.

Image	Tour de France			Graffiti		
Image size	720×480			320×240		
Task/Implement	CPU	GPU	GPU–C	CPU	GPU	GPU–C
Image transfer	N/A	0.727	0.728	N/A	0.174	0.173
Y component	2.683	0.077	0.078	0.718	0.049	0.049
Down sampling	0.633	0.118	0.113	0.131	0.075	0.073
Gaussian filter	221.949	4.057	4.040	53.010	1.860	1.885
ALoG-CD filter	233.024	5.586	5.537	61.803	1.686	1.678
Gradient filter	163.841	2.835	2.802	42.478	0.909	0.899
Feature point	125.680	4.879	4.899	35.220	1.260	1.264
Edge sampling	5.642	5.065	5.097	3.109	1.244	1.245
Orientation map	1243.985	21.128	N/A	248.985	9.311	N/A
Dominant orientation	39.554	1.485	4.243	52.756	0.740	2.434
Descriptor	31.550	2.781	55.005	17.593	1.611	30.978
Total	2069.133	50.174	83.959	516.278	19.683	41.435
#feature	5736			2725		

The details of the computational times. The unit is microsecond.
GPU-C: the GPU implementation without using orientation maps, in which all the pixels in the cells should be accessed to compute the descriptors.