Multimedia HW1

B04902045

November 21, 2017

Results

	Color	Texture	Fusion	RP(25%)	RP(50%)
aloe_vera_gel	0.204	0.080	0.200	0.073	0.084
baby_shoes	0.329	0.168	0.336	0.179	0.182
bicycle	0.199	0.143	0.206	0.164	0.165
bottle	0.300	0.165	0.300	0.139	0.140
bracelet	0.208	0.238	0.230	0.095	0.095
cartoon_purse	0.197	0.086	0.197	0.136	0.143
chair	0.110	0.120	0.114	0.089	0.103
children_dress	0.364	0.152	0.362	0.221	0.269
cup	0.342	0.194	0.349	0.266	0.279
drum	0.227	0.090	0.206	0.115	0.124
garment	0.418	0.375	0.423	0.399	0.380
gge_snack	0.339	0.238	0.359	0.159	0.170
glasses	0.120	0.183	0.123	0.101	0.104
hand_cream	0.270	0.101	0.269	0.179	0.198
korean_snack	0.447	0.094	0.444	0.201	0.207
leather_purse	0.102	0.083	0.105	0.070	0.068
men_clothes	0.332	0.116	0.329	0.218	0.221
minnie_dress	0.530	0.239	0.539	0.213	0.213
minnie_shoes	0.187	0.098	0.188	0.130	0.127
nba_jersey	0.063	0.131	0.071	0.057	0.056
Mean MAP	0.262	0.148	0.264	0.156	0.162

Coding environment and Packages Used

Ubuntu 16.04, Python 3.5.2

- opencv-python (3.3.0.10)
- Pillow (4.3.0)

Note that I uploaded other pdfs for visualization results.

Texture

- **Gabor texture**
- Fourier features
- Co-occurrence matrix metrics
- Tamura's texture
- e.t.c

Implementation

Step 1: Apply gabor filters to every image and calculate mean and variance for each channel(rgb). Texture feature's dimension will be 6(kernel size) x 16(orientation) x 2(mean and variance) x 3(rgb).

Step 2: Normalize every dimension of database feature vector.

Step 3: Compute difference by L1 difference

Color

- **Global color histogram**
- Regional color hisogram
- Grid color moments
- Means in each color channel
- Color (auto-) correlogram
- e.t.c

Implementation

Step 1: Tranform rgb images to HSV space.

Step 2: Quantize them to 288 levels (18 hue * 4 saturation * 4 value).

Step 3: Count the global color histogram and normalize(since pictures differ in size). The normalization here simply means to divide histogram results by the sum of all bars in an image.

Step 4: Compute distance by L1 difference

fusion

Implementation

Step 1: Construct color and texture features.

Step 2: Compute l1 distance separately and normalize both of them.

Step 3: combine them with the weight: 0.9(color), 0.1(texture) and retrieve.

random projection

Implementation

1. Perform random projection on color feature(the projection matrix is sampled from a standard normal distribution)
2. Retrieve by L1 distance

Discussion

- Performing dimension-wise normalization has a negative effect for color features but a positive effect on texture features. The reason to perform dimension-wise normalization is that we want to ensure that all dimensions have the same scale, otherwise certain dimension may dominate the retrieval results. The reason that it doesn't work on hsv is probably because the *hue* dimension(range [0, 360)) is more important than *saturation*(range [0,1]) and *value*(range [0,1]), but performing dimension-wise normalization eliminate this inequality.
- Calculating the result of gabor filters on different channels(rgb) can yield better results(instead of calculating a single mean and variance for a image(after filter is applied) of size [w, h, 3]). But it triples the dimension of the feature vector though, so it can be seen as a trade-off between performance and computation load.
- Random projection seems to lower the MAP pretty much, but it decreases the dimension of the feature vector. Similarly, applying random projection can be seen as a trade-off between performance and computation load. Performance drops dramatically when using hashed features with 50% of the original feature dimensions, but only drops a little(compared to rp50) when I decrease the dimension to 25% of the original feature dimension.
- I also tried to apply a circular shift for gabor features mentioned in `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.8420&rep=rep1&type=pdf`. It doesn't seem to enhance the performance in our case.