
shape_based_matching

update:

fusion implementation to run faster!

icp is also refined to be faster and easier to use

Transforms in shape-based matching

pose refine with icp branch, 0.1-0.5 degree accuracy

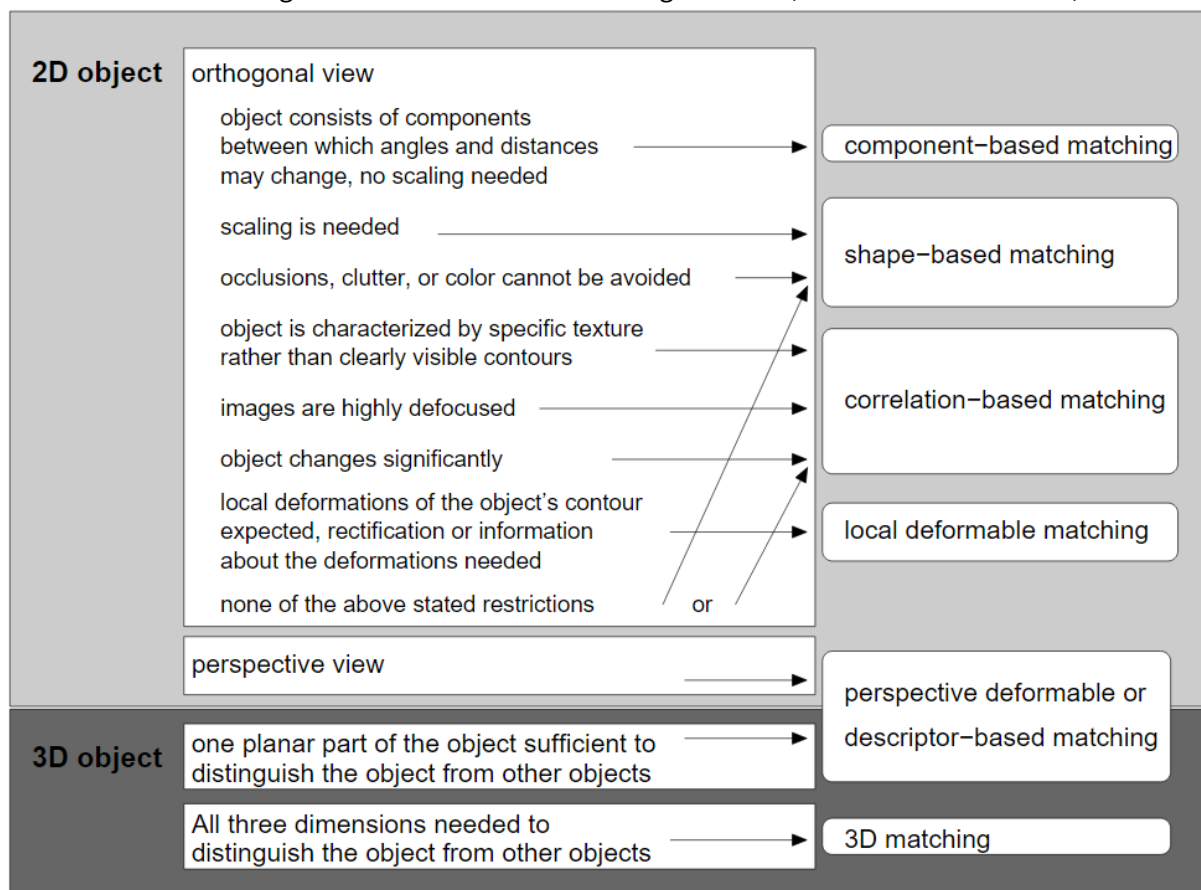
icp + subpixel branch, < 0.1 degree accuracy

icp + subpixel + sim3(previous is so3) branch, deal with scale error

try to implement halcon shape based matching, refer to machine vision algorithms and applications, page 317 3.11.5, written by halcon engineers

We find that shape based matching is the same as linemod. linemod pdf

halcon match solution guide for how to select matching methods(halcon documentation):



steps

1. change test.cpp line 9 prefix to top level folder
2. in cmakeList line 23, change /opt/ros/kinetic to somewhere opencv3 can be found(if opencv3 is installed in default env then don't need to)
3. cmake make & run. To learn usage, see different tests in test.cpp. Particularly, scale_test are fully commented.

NOTE: On windows, it's confirmed that visual studio 17 works fine, but there are some problems with MIPP in vs13. You may want old codes without MIPP: old commit

thoughts about the method

The key of shape based matching, or linemod, is using gradient orientation only. Though both edge and orientation are resistant to disturbance, edge have only 1bit info(there is an edge or not), so it's hard to dig wanted shapes out if there are too many edges, but we have to have as many edges as possible if we want to find all the target shapes. It's quite a dilemma.

However, gradient orientation has much more info than edge, so we can easily match shape orientation in the overwhelming img orientation by template matching across the img.

Speed is also important. Thanks to the speeding up magic in linemod, we can handle 1000 templates in 20ms or so.

Chinese blog about the thoughts

improvement

Comparing to opencv linemod src, we improve from 6 aspects:

1. delete depth modality so we don't need virtual func, this may speed up
2. opencv linemod can't use more than 63 features. Now we can have up to 8191
3. simple codes for rotating and scaling img for training. see test.cpp for examples
4. nms for accurate edge selection
5. one channel orientation extraction to save time, slightly faster for gray img
6. use MIPP for multiple platforms SIMD, for example, x86 SSE AVX, arm neon. To have better performance, we have extended MIPP to uint8_t for some instructions.(Otherwise we can only use half feature points to avoid int8_t overflow)

-
7. rotate features directly to speed up template extractions; selectScatteredFeatures more evenly; exhaustive select all features if not enough rather than abort templates (but features ≤ 4 will abort)

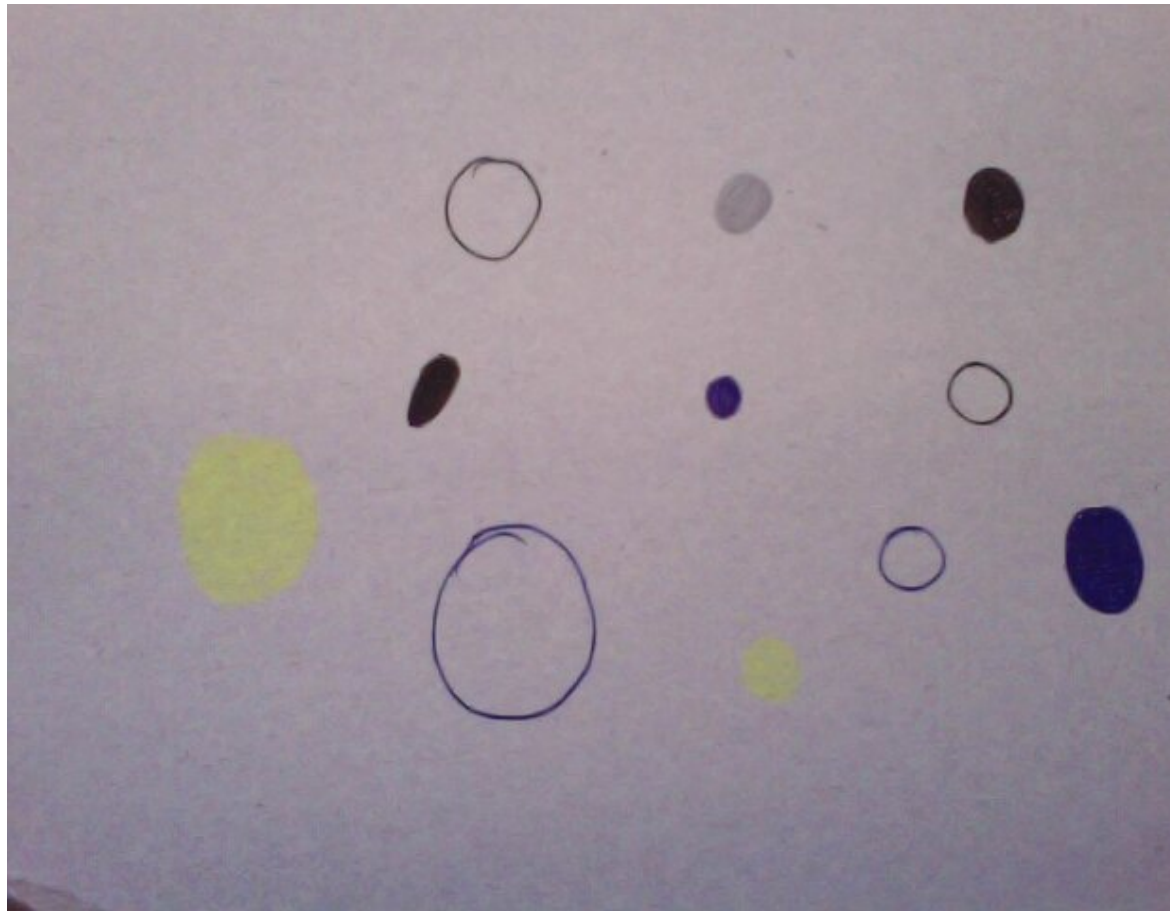
some test

Example for circle shape

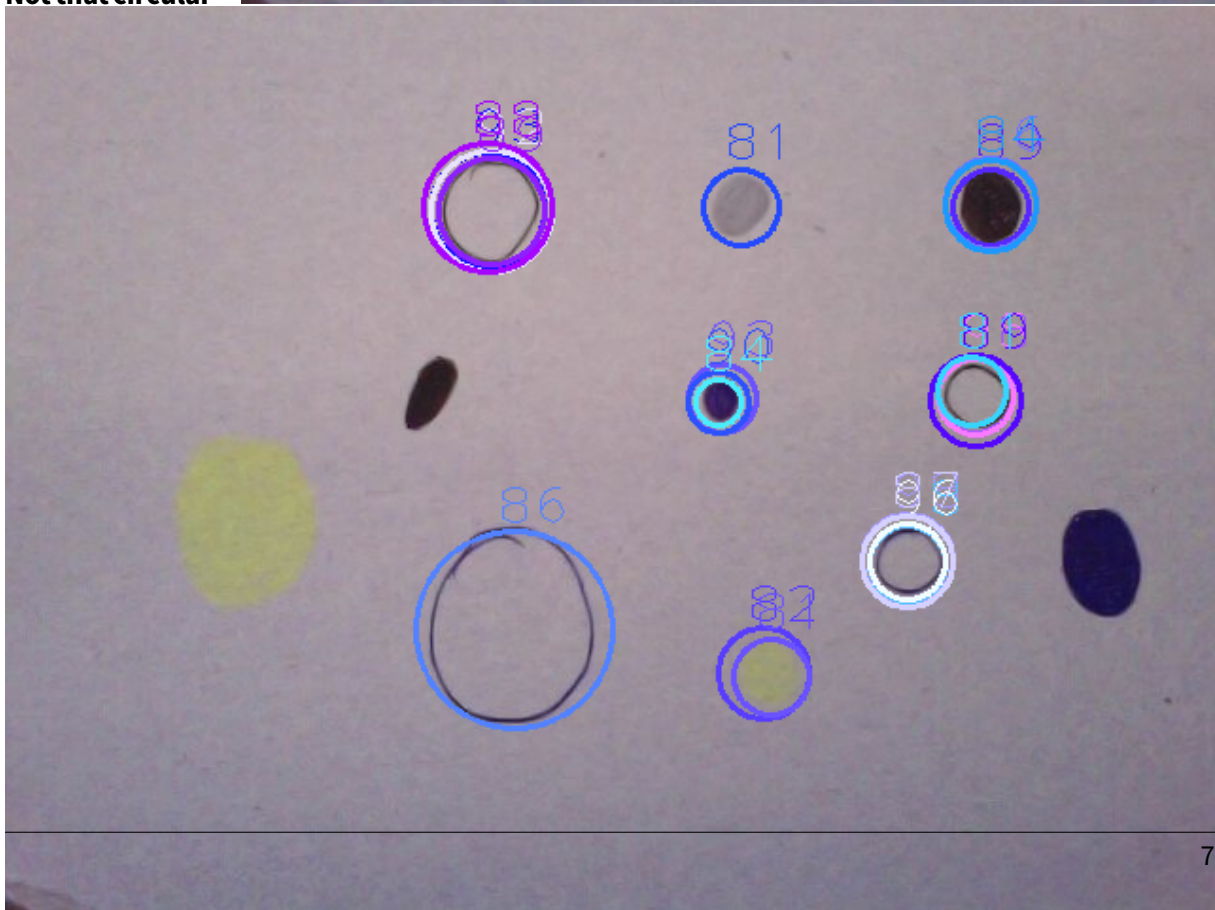


You can imagine how many circles we will find if use edges

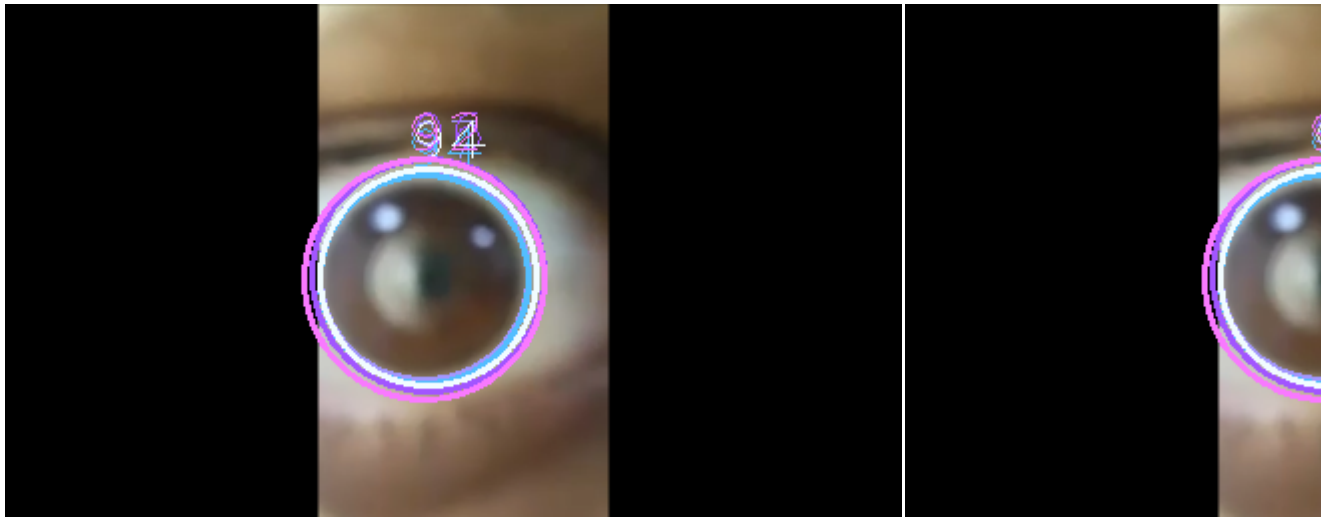




Not that circular

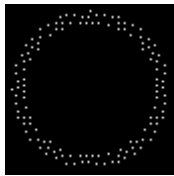


Blur



circle template before and after nms

before nms



after nms

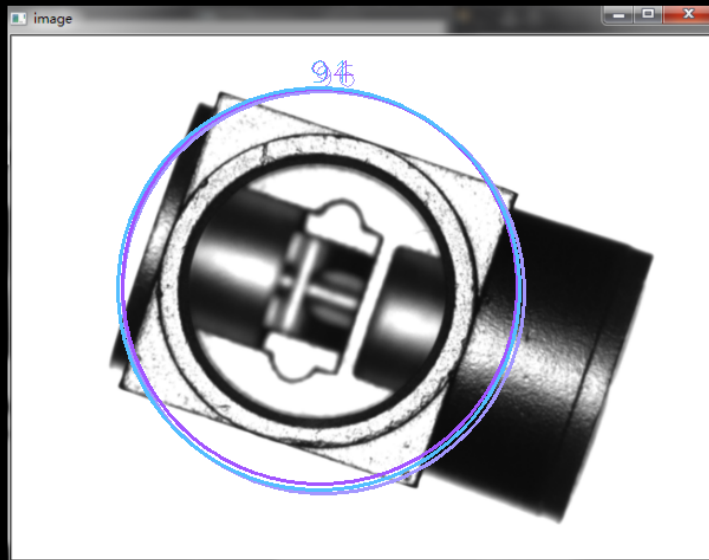


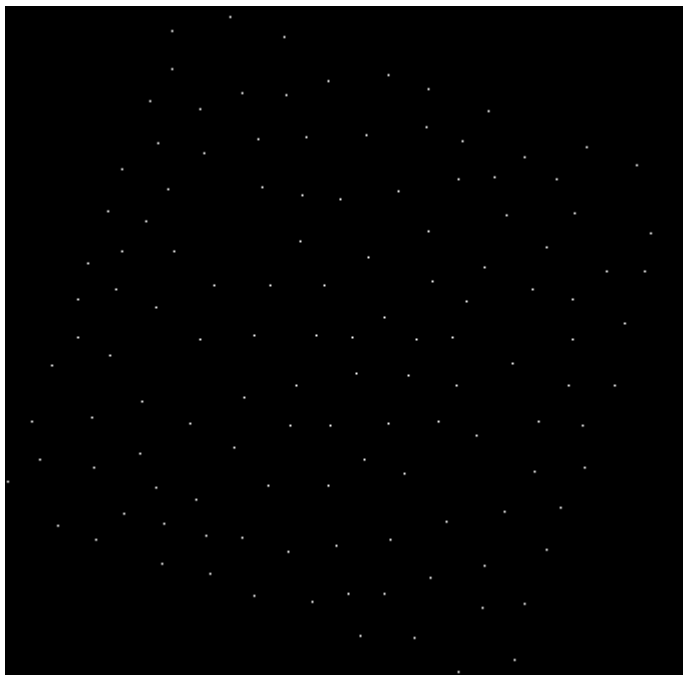
Simple example for arbitrary shape

Well, the example is too simple to show the robustness

running time: 1024x1024, 60ms to construct response map, 7ms for 360 templates

test img & templ features





noise test

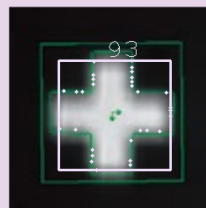
高稳定性 即使拍摄状态与注册时的状态不同，也可正确搜索。



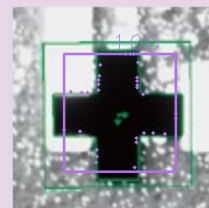
注册图像



缺损



轮廓不清晰



明暗反转

some issues you may want to know

Well, issues are not clearly classified and many questions are discussed in one issue sometimes. For better reference, some typical discussions are pasted here.

object too small?

failure case?

how to run even faster?