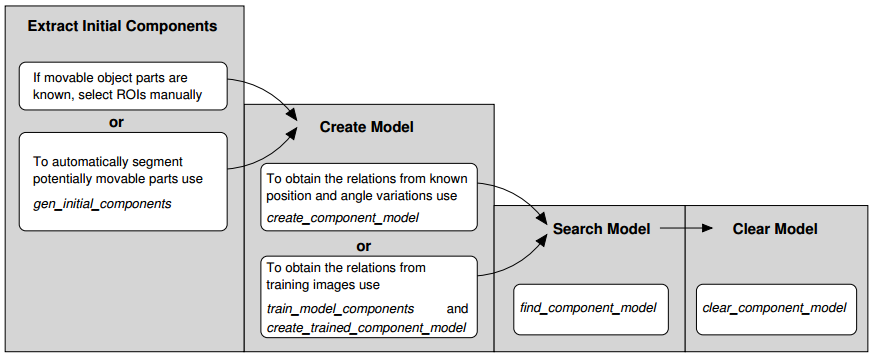
# Component-Based Matching

Component-based matching 是shaped-based matching的扩展。如同shaped-based matching一样，它提取边缘并根据之前建立的模型进行形状匹配。但与shaped-based matching不同，一个部件模型(Component model) 包括多个位置关系可变的部件，例如，它们可以相对移动和旋转。在创建模型时，它们可能的位置关系必须被决定或指定。匹配时返回找到模型实例的各自关系。注意，与shaped-based matching不同，Component –based matching的形状不能有比例变化。

为相对位置关系可变的部件定位比shaped-based matching稍微复杂。例如，在shaped-based matching 中只需要决定一个ROI区域，而在Component –based matching中应该确定包括最初部件（initial components）在内的多个ROI区域。



Component-based matching可分为四步：

1. **提取初始部件**

如果各部件的位置变动范围已知，可以用人为的选择ROI区域，否则可以使用gen\_initial\_components自动对潜在的变动区域进行分割。

1. **创建模型**

使用create\_component\_model根据已知位置、角度变动范围计算得到出部件关系，或者使用train\_model\_components和create\_trained\_component\_model从训练图像中得到部件位置关系。

1. **匹配模型**

使用find\_component\_model来查找和匹配模型。

1. **销毁模型**

使用clear\_component\_model来销毁模型

### 一个简单的实例

例程：hdevelop/Applications/Position-Recognition-2D/cbm\_bin\_switch.hdev

一个开关包括两个相对位置变动的组件，例程定位这两个组件，并根据组件的相对角度关系来确定开关是处于开还是关状态。

**步骤1、提取初始部件**

读取参考图像，图像显示了一个处于开状态的开关，它包括两个相对位置变动的部分，因此创建两个ROI区域，并将其保存在一个tuple数组中（InitialComponents初始组件）。

read\_image (ModelImage, ‘bin\_switch/bin\_switch\_model’)

gen\_rectangle1 (Region1, 78, 196, 190, 359)

gen\_rectangle1 (Sub1, 150, 196, 190, 321)

difference (Region1, Sub1, InitialComponents)

gen\_rectangle1 (Region2, 197, 204, 305, 339)

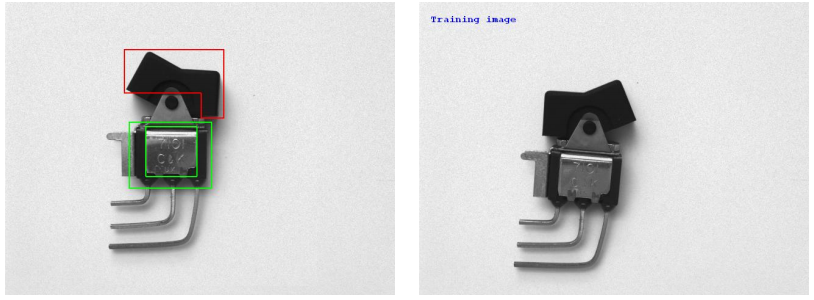
gen\_rectangle1 (Sub2, 205, 232, 285, 314)

difference (Region2, Sub2, InitialComponent)

concat\_obj (InitialComponents, InitialComponent, InitialComponents)

dev\_display (ModelImage)

dev\_display (InitialComponents)



**步骤2、训练组件相对关系**

确定组件相对关系可以采用人为设定的方法，也可以采用训练的方法。这里采用训练的方法。读取一幅训练图像，由于开关只有两个状态，因此，组件间只有两种关系，以初始图像做为参考图像，再读取一幅训练图像便可，如果组件间有多种位置关系，则需要读取多幅训练图像。进行训练的算子是train\_model\_componets，需要的参数有：参考图像、训练图像、和几个控制参数。

read\_image (TrainingImage, ‘bin\_switch/bin\_switch\_training\_1’)

train\_model\_components (ModelImage, InitialComponents, TrainingImage, \

ModelComponents, 30, 30, 20, 0.7, -1, -1, rad(25), \

‘speed’, ‘rigidity’, 0.2, 0.5, ComponentTrainingID)

说明：train\_model\_componets计算训练图像与参考图像的关系，训练结果保存在ComponentTrainingID中，该算子应该用于各部件关系未知的情况，需要自动训练得到，如果部件关系相对位置已知，则不需要训练图像，而直接用create\_component\_model创建组件模型。

**步骤3、创建组件模型**

如果训练结果满意，可以创建组件模型了。由于训练图像只是一个实例，不能覆盖所有的实例情况，因此用modify\_component\_relations来修改模型的容忍度，以使之适应性更强。

modify\_component\_relations (ComponentTrainingID, ‘all’, ‘all’, 1, rad(1))

create\_trained\_component\_model (ComponentTrainingID, 0, rad(360), 10, 0.7, \

‘auto’, ‘auto’, ‘none’, ‘use\_polarity’, \

‘false’, ComponentModelID, RootRanking)

说明：create\_trained\_component\_model与create\_component\_model不同，前者在使用前，必须使用train\_model\_components训练图像，在算子内部建立了多个部件的形状模型，与create\_shape\_model类似。

模型创建后，训练组件不在需要了，用clear\_training\_components算子将其销毁。

clear\_training\_components (ComponentTrainingID)

**步骤4、查找组件模型并提取相应的关系**

现在，可以读取搜索图像使用find\_components\_model进行模型查找和定位了。对于每一个匹配实例，可以采用get\_found\_components\_model来得到组件及其位置关系。

read\_image (SearchImage, ‘bin\_switch/bin\_switch\_’+ImgNo)

find\_component\_model (SearchImage, ComponentModelID, 1, 0, rad(360), 0, 0, \

1, ‘stop\_search’, ‘prune\_branch’, ‘none’, 0.6, \

‘least\_squares’, 0, 0.85, ModelStart, ModelEnd, \

Score, RowComp, ColumnComp, AngleComp, ScoreComp, \

ModelComp)

dev\_display (SearchImage)

for Match := 0 to |ModelStart|-1 by 1

get\_found\_component\_model (FoundComponents, ComponentModelID, \

ModelStart, ModelEnd, RowComp, ColumnComp, \

AngleComp, ScoreComp, ModelComp, Match, \

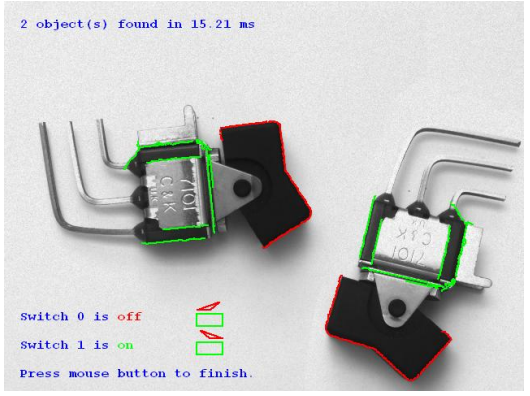
‘false’, RowCompInst, ColumnCompInst, \

AngleCompInst, ScoreCompInst)

dev\_display (FoundComponents)

visualize\_bin\_switch\_match (AngleCompInst, Match, WindowHandle)

endfor



这里的visualize\_bin\_switch\_match是自定义算子，其功能是根据组件的角度关系来确定开关的状态，并将其显示出来。

**步骤5、销毁组件模型**

使用clear\_component\_model 来销毁组件模型。

clear\_component\_model (ComponentModelID)

## 详细说明

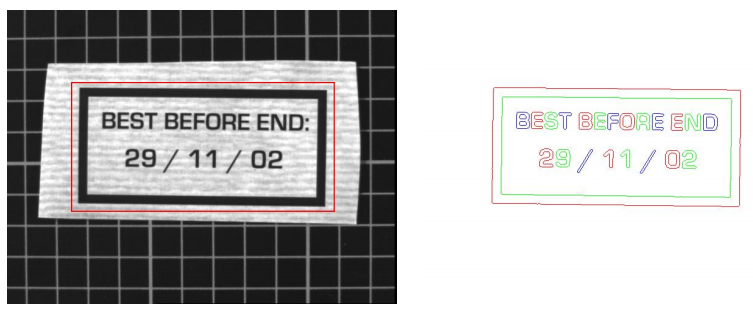
### 一、初始部件的确定

例程：hdevelop/Matching/Component-Based/cbm\_label\_simple.hdev

对于初始组件，即相对位置变化的区域，的位置可以大概确定，可以人为的选择ROI区域，并保存到一个tuple数组中，成为初始组件。

对于初始组件位置未知的情况，可以使用gen\_initial\_components提取。

这里介绍自动提取初始组件的实例。



模板图像是一个印有”BEST BEFORE END”和日期的标签，使用以下代码来打生成初始组件。

read\_image (Image, ‘label/label\_model’)

gen\_rectangle1 (ModelRegion, 119, 106, 330, 537)

reduce\_domain (Image, ModelRegion, ModelImage)

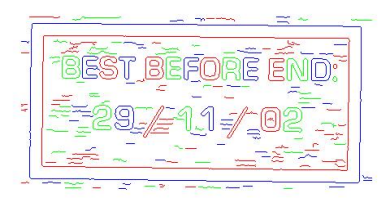
gen\_initial\_components (ModelImage, InitialComponents, 40, 40, 20, \

‘connection’, [], [])

dev\_display (Image)

dev\_display (InitialComponents)

关键算子是gen\_initial\_components，参数40，40，20表示提取轮廓的最小最大对比度和连接的轮廓最最小的间隔，调整这三个参数，使得提取出来的轮廓尽可能的完整且不被无关的纹理干扰，例如，不要发生这种情况：



当初始组件从参考图像中提取出来，接下来就可以创建组件模型了。组件模型可以采用不同方法创建。

### 创建合适的部件模型

**如果可能的相对关系已知，可以直接使用create\_component\_model来创建。**

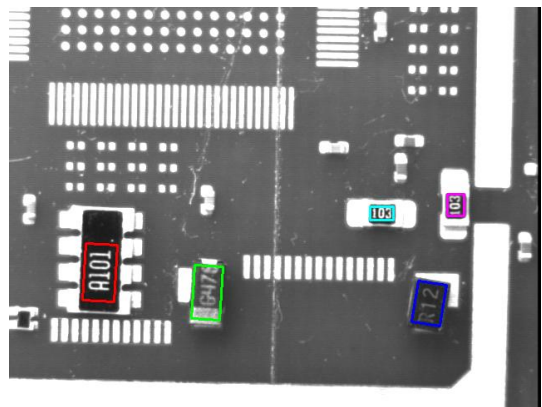
**如果可能关系未知，可以能过一系列训练图像来计算所有可能的位置关系。**

如果采用训练的方法创建模型，在训练之后和模型创建之前，可能需要查看不同的训练结果或者修改训练结果。

模型建立之后，你可能想要保存模型到文件，以便在其它应用中使用，或者从已存在的模型中获得信息。

**建立相对未知已知的模型**

对于相对位置移动已知的模型组件，可以直接使用create\_component\_model来创建组件模型，实例见：hdevelop/Matching/Component-Based/cbm\_modules\_simple.hdev。



read\_image (ModelImage, ‘modules/modules\_model’)

gen\_rectangle2 (ComponentRegions, 318, 109, -1.62, 34, 19)

gen\_rectangle2 (Rectangle2, 342, 238, -1.63, 32, 17)

gen\_rectangle2 (Rectangle3, 355, 505, 1.41, 25, 17)

gen\_rectangle2 (Rectangle4, 247, 448, 0, 14, 8)

gen\_rectangle2 (Rectangle5, 237, 537, -1.57, 13, 10)

concat\_obj (ComponentRegions, Rectangle2, ComponentRegions)

concat\_obj (ComponentRegions, Rectangle3, ComponentRegions)

concat\_obj (ComponentRegions, Rectangle4, ComponentRegions)

concat\_obj (ComponentRegions, Rectangle5, ComponentRegions)

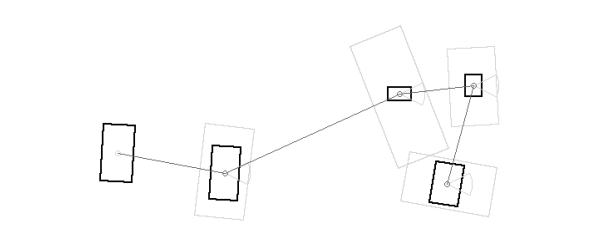
create\_component\_model (ModelImage, ComponentRegions, 20, 20, rad(25), 0, \

rad(360), 15, 40, 15, 10, 0.8, [4,3,3,3,3], 0, \

‘none’, ‘use\_polarity’, ‘true’, ComponentModelID, \

RootRanking)

利用get\_component\_model\_tree可以获得关系树各部件关系，关系树指定了一系列部件和它们之间是如何进行搜索的，这取决于选择哪一部件作为根部件，根组件是首先搜索的部件，参考部件是用来计算各部件中关系的组件，这是部件的前驱部件。

 get\_component\_relation主要用来估计在创建部件模型前训练是否成功，而get\_component\_model\_tree用于在创建模型之后，可视化搜索空间。

关系树中：

1. 各部件参考点的位置以小圆表示，相应的坐标以对数Column和Row返回。
2. 各部件除根部件外，相对于各参考部件的角度变化量以固定半径的扇形表示。
3. 除根部件外，所有部件的位置关系，指定部件相对于参考部件的参考点的移动量用矩形表示。

如果各部件的关系未知，你需要使用训练图像来得到它们的相对关系。

例程：hdevelop/Matching/ComponentBased/cbm\_label\_simple.hdev

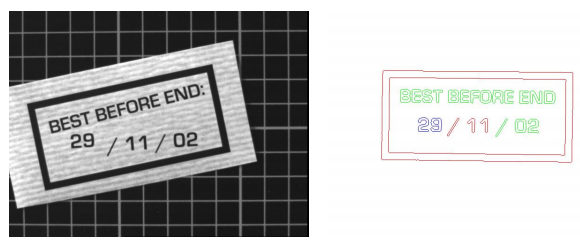
train\_model\_components (ModelImage, InitialComponents, TrainingImages, \

ModelComponents, 40, 40, 20, 0.85, -1, -1, rad(15), \

‘speed’, ‘rigidity’, 0.2, 0.5, ComponentTrainingID)

dev\_display (Image)

dev\_display (ModelComponents)



由结果可见，初始模型中相对位置不变的组件被合并成为一个组件。

训练完成之后，可以使用create\_trained\_component\_model来创建组件模型。在创建以前，使用modify\_component\_relations为模型增加一点容忍度。在模型创建这后，训练组件不经需要，可以销毁。

modify\_component\_relations (ComponentTrainingID, ‘all’, ‘all’, 15, rad(5))

create\_trained\_component\_model (ComponentTrainingID, -rad(30), rad(60), 10, \

0.8, ‘auto’, ‘auto’, ‘none’, \

‘use\_polarity’, ‘false’, ComponentModelID, \

RootRanking)

clear\_training\_components (ComponentTrainingID)

注意，gen\_initial\_components可以用来测试在不同的ContrastLow，ContrastHigh，MinSize值的情况下得到的模型组件。因此，对于训练组件模型，这一操作类型于shape-based matching中的inspect\_shape\_model。