Guidelines for installation and usage of vision system for pad printing machine

Setup

For the instructions of setting up the environment, please refer to the setup documentation. This document assumes a working installation of a python compiler (Anaconda or the like). Further, install OpenCV following the same document.

Prerequisites

Start this tutorial only after the completion of basics of openCV and reading other documentations like SAP integration. Also one needs to understand working with video using openCV[4].

Softwares

OpenCV - Image processing

Trails result



Figure : Working demo of the poka-yoke

Mechanical installation

The choice of installation of the web cameras is shown in the images. Camera-1 checks the orientation of the rim and is placed just outside the change plate assembly as shown in the image. Camera-2 checks the current model of the rim and is placed vertically above the change plate.



Figure : Rim seen from the top view

Codes

There are two requirements for the working of this poka-yoke. Two different functionalities are implemented as a set of codes for the model identification and the other set for orientation check. Both of these functionalities are implemented in the camutils.py file.

Orientation check:

The orientation of the placed rim is verified by identifying the text present on the inner surface of the rim as shown in the figure.

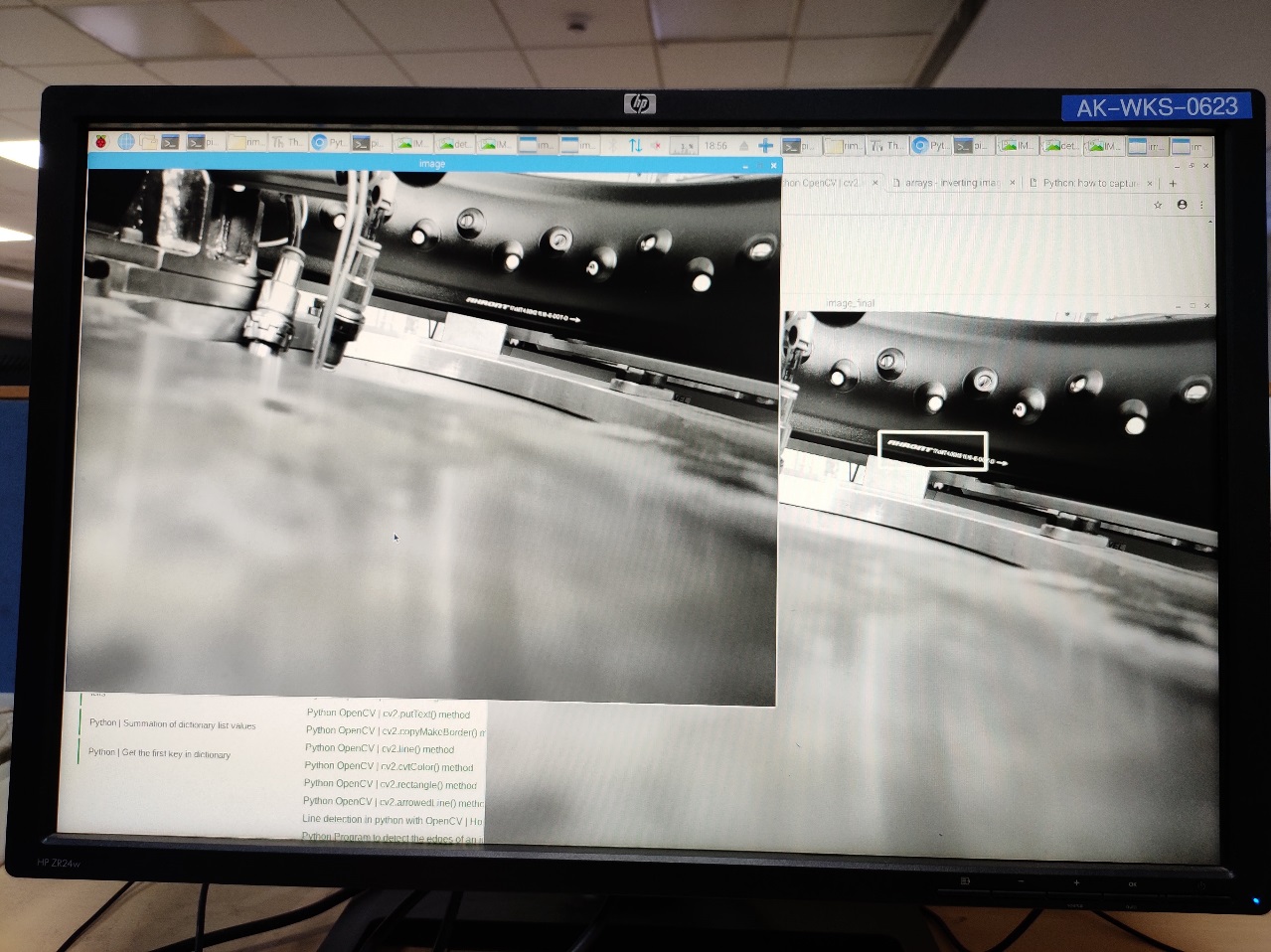


Figure 3: Orientation check poka-yoke detecting the text

Since this setup does not have an external light source, it is prone to variations from the external lighting. So the final code is complete only while actually deploying the system. The orientation check uses a technique called blob detection to identify the text. Instead of using many in-built functionalities we will write a simple blob identification algorithm to solve the problem.

Once the image is captured from the camera, use thresholding[1] to convert it into a binary image. Adjust the threshold value to suppress any noise in the image and only the text to be identified. If required crop the respective position in the image for this step.

Once we have a thresholded image, use morphological transformations like dilation[5] to join the identified text into once single white patch or blob. Now, we have a thresholded image with a large patch of white with black background. Use contour identification[2] to identify the patch and then draw a boundary[2] around the identified region. The area of this region is an indicator for the correct orientation of the rim placed.

Model identification check:

Any given model is identified based on a simple template matching technique[3]. Once the camera is installed on the setup, the images of different models are captured and are stored in memory. Whenever Camera-2 gets a trigger, the current image captured is compared with all the templates and the relative scores are compared. The maximum score obtained will be the model present. A similar example for cylinder head identification is given below.

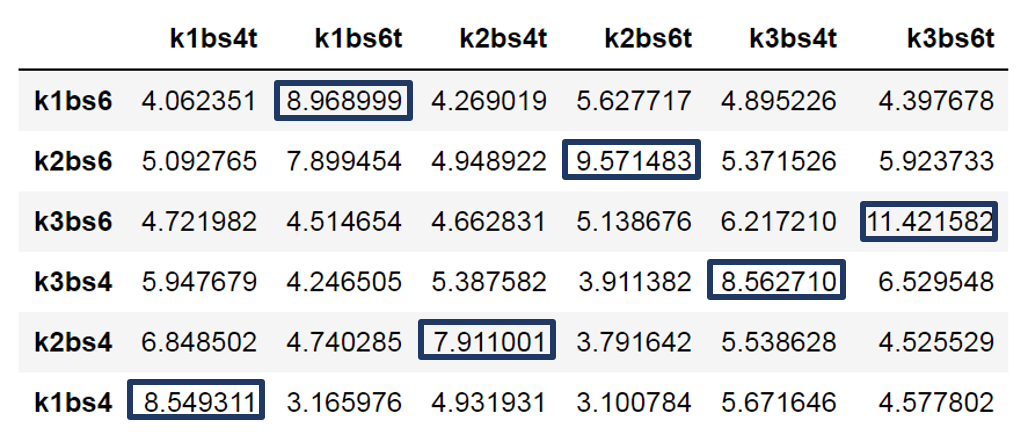


Figure : Scores when matching template vs actual image

In the above table the top row is the template images of the respective actual images given in the first column. As one can see the maximum matching score is obtained when the corresponding images are matched with their templates.

Common problems

Remarks

Please write to [akhilsathuluri@gmail.com](mailto:akhilsathuluri@gmail.com) for further queries.

References

[1] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html>

[2] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html>

[3] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html>

[4] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_video_display/py_video_display.html>

[5] <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html>