



 Popular





TOPICS


 Internet Culture (Viral)

 Games


 Q&As

 Technology


 Pop Culture


 Movies & TV


See more





RESOURCES


 About Reddit


 Advertise


 Help


 Blog


 Careers


 Press


 Communities


 Best of Reddit



 Topics

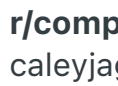
 Impressum


 Content Policy

 Privacy Policy

 User Agreement

  r/computervision · 4 yr. ago

 caleyjag



Python library for industrial image processing?

Python

In my normal day job I use industrial image processing packages like Cognex Vision Pro, MVTec Halcon and the LabVIEW Vision Dev Module (which is a personal favourite.)


These libraries have a lot of functions that are specifically tailored for the kinds of vision applications you would expect to see on a factory line, where precise pass/fail metrics and decisions are needed.


When tinkering in Python I usually use OpenCV and scikit-image but as far as I can tell these libraries tend not to overlap too much with the industrial ones I mentioned above. Of course that is perfectly understandable since they need to be more general.


For example, a common tool in an industrial library would be a rake function to find straight edges with sub-pixel precision. As far as I can see this sort of tool is generally absent in OpenCV and the like (although I may not be looking in the right place).


Are there any Python libraries (or repositories) that have a bit more of an industrial flavour to them?


As a last resort I can embed Halcon or VisionPro functions within my Python solutions but that comes with licensing cost implications as well as the burden of additional runtime environments.


 39



 18




 Share

 Report

Single comment thread

[See full discussion](#)

 Legal_Ride_638 · 4y ago

I experienced the same thing as you with OpenCV. My work experience is in industry using libraries from Cognex, MVTec, Fanuc, Keyence etc. A couple years ago I went back to school to do my Master's in computer vision and I was surprised at the state of affairs. Everyone was using Matlab or OpenCV. At first I thought OpenCV was neat, but I quickly realized that it did not have the capability to perform the type of applications we do in industry. Not even close. I would say that the Halcon library in particular makes OpenCV look like a children's playground when it comes to industrial inspections. Even in regular computer vision tasks I would most often prefer Halcon. I could literally go around the computer vision lab at the school and complete applications in hours that graduate students had been trying to do in OpenCV for months. No one had any idea that industrial libraries even existed.

I can't speak for LabView since I never used it. As some other people pointed out, most of the industrial software contains high level tools and the user doesn't really need to know much. I think the exception would be Halcon. It contains most of the low level tools (sobel, hough, canny, etc) as well as many high level tools (matching, camera calibration, PnP etc). OpenCV also contains low level tools (sobel, pyramid, etc) as well as high level tools (SIFT, AKAze, PnP, camera calibration etc). I have used both of these libraries extensively and they are my favorites.

I would never use OpenCV for industrial machine vision applications. In my opinion it is no where near as good as the industrial ones. Since I can hear the OpenCV fan boys perking up I have decided to make a list of reasons. Here I will compare Halcon to OpenCV since they are probably the closest in terms of functionality. I would consider both of them general purpose vision libraries.

1) Matching - The industrial pattern matching algorithms are by far the largest technical advantage over OpenCV. The algorithms are almost all focused on objects with no texture. Most of them rely on edge gradients as the primary feature for matching. The main algorithms for feature matching in OpenCV are SIFT, SURF, ORB etc and these are completely useless for texture-less objects. OpenCV does have a generalized hough transform but the implementation is not invariant to rotation, scale, perspective, local distortion etc. OpenCV also has LineMod in contirb but it does not achieve 1/22 pixel accuracy like the methods in Halcon and the implementation expects a kinect image.

The same thing is happening in the 3D world. Here is a link to the leader board of the latest BOP challenge(<https://bop.felk.cvut.cz/leaderboards/>). Notice that most algorithms are not performing well on the ITODD dataset (industrial parts). The algorithms that do perform well (top 5 at this time) are those based on point pair features which is patented by MVTec (makers of Halcon).

Here is a list of the top matching algorithms found in Halcon, all of them are patented:

1) Shape Based Matching (uses edge gradients):

2) Compound Parts Matching (example - scissors)

3) Perspective Deformable Matching (3D pose from single image of planar objects):

4) Local Deformable Matching (example - chip bag)

5) Shape Based 3D Matching (3D pose in single image using a CAD model):

6) Surface Based Matching (point cloud matching):

7) Surface Based Deformable Matching (deformable point cloud matching)

8) Surface Based with Images (point cloud combined with 2D images)

Beyond these Halcon also has rotation invariant correlation based matching and 3 descriptor based matchers.

Algorithms 1-8 above are going to destroy anything you can find in OpenCV in terms of accuracy, performance, documentation and ease of use. OpenCV just can't do what these algorithms do at all. And even if you spent months trying to implement one of these using the tools found in OpenCV, they are patented so you would have to pay royalties to MVTec anyways. Also, these algorithms cannot be implemented as easily with basic functions in OpenCV as some people think. Many of the OpenCV functions would have to be pulled apart ,re-purposed or completely re-written. You could not simply just call a sequential set of OpenCV instructions and implement those algorithms and get the same performance as in Halcon.

2) Image Acquisition - Halcon has support for all the major image acquisition devices (GigE, USB3, directshow, GenICam etc). I can purchase cameras from Baumer, Basler, Allied Vision, SensoSo, LMI etc and just plug them into my computer and start capturing images. OpenCV does not have support for any of the major acquisition interfaces.

3) Deep Learning - Halcon has classification, object detection, object detection with rotation, semantic segmentation and anomaly detection. And best of all, I can actually train the networks using Halcon (without a GPU). OpenCV does not have the ability to train convolutional neural nets at all. The process for OpenCV is to first train one using PyTorch/Tensorflow/Keras, then try and convert that network to something that OpenCV can read and perform inference on. What could possibly go wrong.

4) 3D Vision - Another area where Halcon absolutely obliterates OpenCV. Halcon has multiple methods for obtaining the 3D pose of an object from a single image including CAD matching, Perspective Deformable , Descriptor based and PnP. OpenCV only has PnP. Halcon can reconstruct a 3D scene from laser triangulation, stereo vision, multi-view stereo, depth from focus and photometric stereo. OpenCV only has stereo vision. Halcon has the ability to perform surface matching in a 3D scene to obtain an objects pose. It also has the ability to perform operations/measurements on point clouds such as segmentation, thresholding, smoothing, moments, areas, volumes etc. OpenCV has nothing for dealing with point clouds.

4) Calibration - Halcon supports calibration routines for area scan camera, line-scan cameras, tilt lenses, telecentric lenses, and hypercentric lenses. OpenCV supports area scan only.






The list could go on and on. Halcon supports deflectometry, sub-pixel edge detection, efficient region processing, code reading, OCR, measuring tools, sockets, RS-232, industrial communication protocols and a ton of other things not found in OpenCV.


I can hear people crying foul now due to the high cost of the industrial libraries. Last time I checked Halcon is over \$7000 for a development license over \$2000 for a run-time license. Most people gasp at these numbers. However if you really have the skill to implement the algorithms listed above (you are a computer vision expert) then you should value your time at at least \$100/hr. Now start coding up just one of those algorithms listed above and let me know when you get an implementation that matches Halcon's. You will be at \$7000 in just 70 hours (less than two weeks).

When I was in graduate school I couldn't believe how much time students wasted on computer vision. They would have a two year project performing robot guidance that would take us 2-3 weeks in industry. And it's all because they were trying to use OpenCV and Matlab.

Here are cases where I would use/do use OpenCV:



- In a research setting where you are trying to implement a more efficient algorithm than one found in OpenCV. Or you need to dissect/re-purpose an existing algorithm only found in OpenCV.
- If you have a specific product that will sell hundreds/thousands/millions of copies. At some point, it makes sense to sit down for a long time and code up an algorithm and not have to pay a run-time license of \$2000/unit.
- If you are a hobbyist and only do computer vision for fun. I still love using using OpenCV at home, but for my real job I prefer industrial libraries.
- The cost of downtime was non-existent. Downtime in large factories can easily be in the thousands of dollars per minute. When I use an industrial library, I know it is well tested and there is a good support system if it goes down. What are you going to do if OpenCV crashes and shuts the production line down? Get on stack overflow and wait to talk to some kid sitting in his parents basement? If downtime was not an issue, I would be more inclined to use OpenCV at least for some simple applications.




  11  Reply  Award  Share ...

 caleyjag OP · 4y ago

Superb reply. This could be a document to explain to management why they need to pay out for a commercial package license!

Thanks for the sanity check.

 1 

 Reply  Award  Share ...

r/computervision

Join

Computer Vision

Computer Vision is the scientific subfield of AI concerned with developing algorithms to extract meaningful...

Show more

96K

Members

20

Online


Top 2%

Rank by size

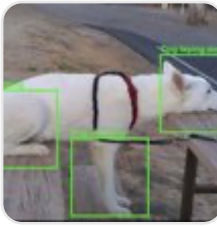
Related

Computer


Information &

 r/computervision

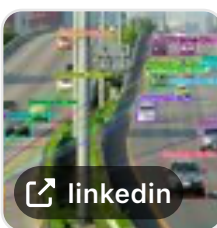
Can language models help me fix such issues in CNN base...




429 upvotes · 60 comments

 r/computervision

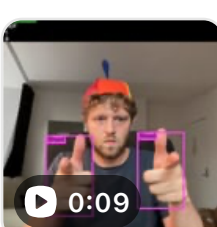
Ultralytics making zero effort pretending that their code...




106 upvotes · 68 comments

 r/computervision

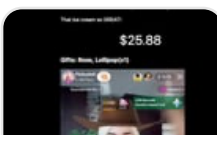
I want to test my inference model. Anyone up for a live...



108 upvotes · 10 comments

 r/computervision

I trained a model on all Tiktok



Reddit, Inc. © 2024. All rights reserved.