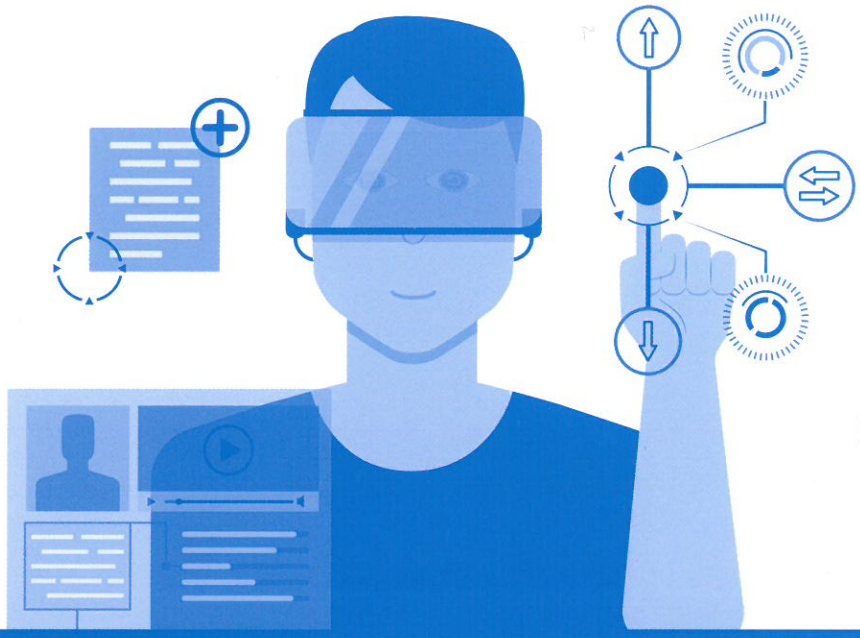
Patrick Jost, Guido Kempter (Hrsg.)



uDay XVI

Beiträge zum Usability Day XVI

Assistenztechnologien in der Arbeitswelt

21. Juni 2018

PABST

Kontakt
Patrick Jost, BSc MA PGCert
University of Applied Sciences Vorarlberg
UCT Research
Hochschulstraße 1
6850 Dornbirn
Österreich
E-Mail: patrick.jost@fhv.at

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Druck: KM-Druck, D-64823 Groß Umstadt

Print: ISBN 978-3-95853-405-6
eBook: ISBN 978-3-95853-406-3 (www.ciando.com)

Inhaltsverzeichnis

Ryan, R. M., & Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. Contemporary Educational Psychology, 25(1), 54–67. doi: 10.1006/ceps.1999.1020.

Salmi, H., Thuneberg, H. & Vainikainen, M.-P. (2017). Making the invisible observable by Augmented Reality in informal science education context. International Journal of Science Education, Part B, 7(3), 253-268. doi: 10.1080/21548455.2016.1254358.

Schart, D., & Tschanz, N. (2015). Augmented Reality (1. Aufl.). Konstanz: UVK.

Sherman, W. R., & Craig, A. B. (2003). Understanding Virtual Reality. Interface, Appli-cation, and Design (The Morgan Kaufmann series in computer graphics and geometric modeling, 1. Edition). San Francisco: Morgan Kaufmann.

Vester, F. (2016). Denken, Lernen, Vergessen. Was geht in unserem Kopf vor, wie lernt das Gehirn, und wann lässt es uns im Stich? (37. Auflage). München: Dt. Taschenbuch-Verl.

Detecting and Locating People Using Low-Cost Thermal Imaging Camera

Michal Charvát & Martin Drahanský

Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Czech Republic

Abstract. The paper is a short version of a bachelor's thesis, which will be available later this year. As we are approaching times of Industry 4.0 and smart homes, there is a need for automatic reliable mechanism for detecting living beings and there are many use cases—from aiding elderly people with their everyday life to increasing safety in dangerous workplaces by guarding hazardous areas. We propose a method for counting and locating people using single low-cost thermal camera module and Raspberry Pi-like computer. This paper describes the process of necessary hardware configuration of the whole system and a software solution to the problem of detecting and locating people, which is based on applying human temperature filter, image processing with object detection using OpenCV and 3D scene reconstruction with known environment parameters. This approach – even in its simplest form – provides accuracy over 90 % on our data set with various possibilities for improvement.

1. Motivation

Detecting and locating people has found its usage in many areas of everyday life. It is used for queue management in shops and often in marketing for determining best product placement based on a model constructed with data of customer movement, where they spend majority of time, what path they tend to take etc. Another usage is in smart homes. Having information about people's presence, location and/or pose, we can control their environment to ease their everyday life such as helping elderly people with turning machines on or disabling them when there is a high chance of them forgetting to do so. One of the most important areas, where living beings detection is used, is undoubtedly workplace safety mechanisms—hazardous areas guarding at train stations, in heavy machinery and industry halls.

For purposes of detecting / locating people several technologies are nowadays being used:

- Infrared/laser beam interruption for counting people,
- Laser light burst travel time for counting people,
- 3D stereo video analysis,
- Projecting structured light and image analysis for depth vision,
- GPS / WiFi / Bluetooth smart devices tracking,
- Monocular video analysis.

Using thermal imaging camera module to help solving this problem in real world situations belongs to the monocular video analysis section and brings several advantages when compared with other approaches. To begin, we need only a single camera: there is no need for

extremely precise calibration as with stereo vision or structured light projection. Furthermore, we can detect but also locate living beings. Infrared/laser beam or light travel are usually used only for count objects entering and leaving the room. The biggest advantage however, is that by using thermal camera it is impossible to perform facial or person recognition. This makes this approach more suitable for places where privacy plays an important role like workplaces or homes.

2. Hardware part of the system

This chapter describes all hardware parts of the project along with a necessary procedure on how to set up the whole system to be able to receive frames.

2.1 Thermal camera module

In this project we use a Lepton® 3 thermal camera module made by the company FLIR. The LWIR camera module is smaller than a dime and provides decent images with its 160 by 120 pixels resolution. Captured frames are transferred in either RGB888 (24bit) or Y14 raw gray-scale format. Lepton camera module communicates via two interfaces. It uses I^2C interface for receiving commands from the master device and 20 MHz SPI for transferring video frames.

2.2 Master controlling device Orange Pi

In order to be able to communicate with the module we had to choose an appropriate low level computer that is also powerful enough to operate SPI interface running on relatively high frequency 20 MHz. This fact or a necessary video buffer size (38.4 kB for a single video frame) disqualified many well-known boards like the Arduino (max 20 MHz clock speed but only 2 kB RAM). It seemed only reasonable to choose single-board computer of a Raspberry Pi type.

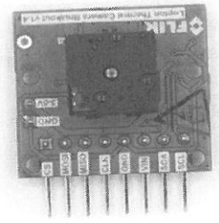


Illustration 1. Lepton 3 thermal camera module with breakout board

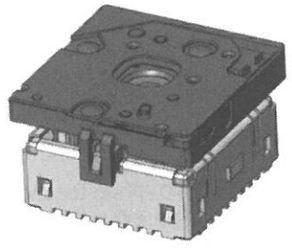


Illustration 2. Lepton 3 thermal module

or this project we have chosen an Orange Pi PC2[1] running Armbian OS[2] (Ubuntu 16.04 based), which is not only a cheaper but also a more powerful brother of the Raspberry Pi. The board is running quad-core 64-bit Cortex A53 CPU with 1 GB RAM and as usual with Raspberry Pi-like boards it has 40 general IO pins connected to various hardware modules. In this project we are interested in SPI and I^2C hardware modules that enable us to control and receive video frames from the thermal camera directly from the high level UNIX system run by the processor. The CPU has also enough processing power to maintain smooth operation without delays. This turned out to be crucial for maintaining synchronization with the camera module when transferring video frames.

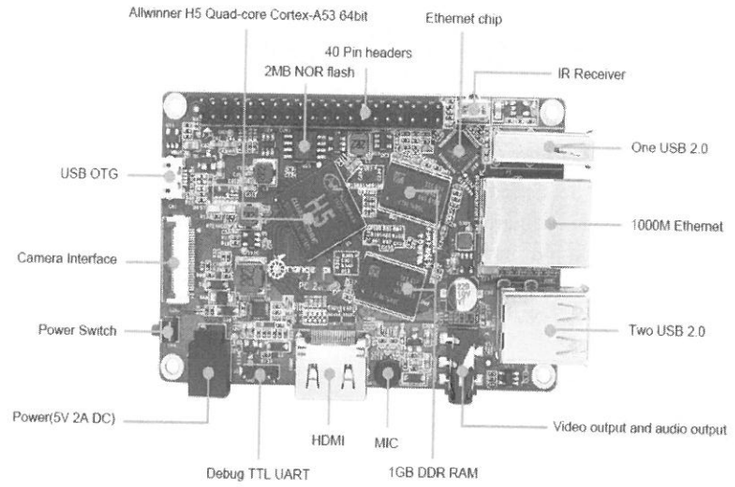


Illustration 3. OrangePi PC2 computer overview, source: (official website) http://www.orangepi.org/orangepipc2/images/orangepipc2_info.jpg

2.3 Enabling hardware modules on Orange Pi

Most of the times operating systems that are available for Raspberry Pi-like boards come with hardware modules like I^2C or SPI disabled by default and pins that are connected to them are used as standard IO pins. One can access these hardware modules on UNIX running machines using so called devices in `/dev/*` directory and control them with standard C calls like `ioctl()`. If there is not a device for a particular hardware module listed in the `/dev/*` directory, then the module is disabled.

1 Orange Pi PC2 official site http://www.orangepi.org/orangepipc2/

2 Armbian OS for OrangePi PC2 https://www.armbian.com/orange-pi-pc2/

Illustration 4. Listed devices corresponding to SPI and I2C hardware modules

Enabling these modules happens not to be a simple task since there is no officially documented straightforward procedure. Steps required to enable them vary from one operating system to another. Bits and pieces on how to enable the modules for Armbian operating system had to be collected throughout multiple Armbian forums.

As was mentioned in previous paragraphs – a device tree (DT) is a way of describing hardware module configuration to the operating system kernel. In order to enable hardware modules of the processor we need modify the device tree. This can be done using so called device tree overlays – compiled configuration files loaded at startup and used by the kernel to create needed hardware module pin mapping. The Armbian OS comes with several DT overlays already included. First step then is to only use the provided DT overlay at startup. This can be achieved by modifying the /boot/armbianEnv.txt:

```
overlays=spi-spidev
param_spidev_spi_bus=0
param_spidev_max_freq=100000000
```

This will use a provided precompiled SPI device tree overlay for mapping the /dev/spidev0.0 device into the system DT. In case there is no such device visible in the DT after rebooting, that could mean that the provided precompiled DT overlay spi-spidev is incorrectly configured. In that case one must take the longer way of decompiling the provided overlay first, reconfiguring it, compiling it back and only then will the device hopefully appear in the DT. Compiled overlay files are stored in the /boot/dtb/allwinner/overlay/ directory.

They can be decompiled to a text form using device tree compiler:

```
$ dtc -I dtb -O dts -o cpu-prefix-spidev.dts cpu-prefix-spidev.dtbo
```

Configured and recompiled:

```
$ dtc -I dts -O dtb -o cpu-prefix-spidev.dtbo  cpu-prefix-spidev.dts
```

In this case the only thing needed to be changed in the overlay was „status“ of the module to „okay”.

3. Camera communication software library

Once the proper functionality of both I²C and SPI interface was confirmed using an Arduino board as a loop-back device, next step was to develop a software library for controlling the camera and for capturing frames. Unfortunately, none of the ready-made solutions found online would work for the newer Lepton 3 camera because either they would be incomplete or programmed for previous version of the camera – Lepton 2 whose VoSPI (video over SPI) protocol is not compatible with the newer one that we are using. Besides, many existing libraries are pointing out to problems with losing synchronization with the camera. Since the camera requires quite fast SPI transmission (20 MHz), it is necessary to use short wiring – no longer than 15 cm. If possible – use additional capacitors and pull-up resistor on the SCK line close to the camera and most importantly, use highly efficient code that is able to process incoming data with minimal delay. As a part of the project, we introduce v4l2lepton3 library written in fast C++ with dual packet buffering that works in hand with a virtual video loopback device v4l2loopback[3]. The v4l2lepton3 library writes frames from a Lepton 3 camera module into a virtual video device, which then provides accessibility to the video stream for all standard UNIX tools like *ffmpeg, gstream, vlc.* The video can also be streamed over a network. Capturing library supports both RGB888 and Y14 formats and its implementation allows for easy frame post-processing extension only by overriding a single method. One of its major properties is speed – it does not lose synchronization with the camera while transferring video frames.

Illustration 5. Lepton 3 video frame captured by v4l2lepton3 in raw Y14 format after linear normalization

Illustration 6 (colored). Lepton 3 video frame captured by v4l2lepton3 in RGB888 format colored using fusion palette after internal histogram image normalization

The library is publicly available in a git repository[4] and also contains lepton3.py script that allows for controlling the camera using commands. The script uses smbus2[5] python library to access I²C device in the system DT and contains prepared infrastructure for sequential reading/writing and setting random registers in the camera. This way it is fairly easy to add

3 Official v4l2loopback virtual video device git repository https://github.com/umlaeute/v4l2loopback

4 Official v4l2lepton3 library git repository https://gitlab.com/CharvN/v4l2lepton3

5 Smbus2 python library for controlling I2C devices https://github.com/kplindegaard/smbus2

not implemented commands from the documentation. Commands sent to the camera must strictly follow the communication protocol described in the documentation. At this moment, control script implements commands for controlling histogram image optimization (AGC), changing output format, color palette, enabling radiometry etc.

With the use of v4l2lepton3 library, we are able to capture frames, video from the Lepton 3 camera module and also control it using the lepton3.py script.

4. Image processing and object detection

This section describes a method of detecting people from a single raw thermal image and a technique used for 3D scene reconstruction used in this project.

4.1 Pixel values to temperature

Once we have obtained a single image in raw format (Y14 gray scale), whose pixels are unaltered values of flux coming to the sensor, we must create a mapping function between flux and an actual real temperature. Although the Lepton 3 module has radiometry support, it is not in fact true radiometry in a sense that we would be able to receive temperature readings from the sensor directly. Its radiometry mode only compensates for change in internal and ambient temperature difference so that its raw output does not get affected by this change of temperature and always produces same data with respect to incoming infrared flux. With this kind of camera, it is convenient to use an external spot thermometer device for calibration purposes such as the MLX90614[6]. Using this device, we can create a precise mapping function between flux values produces by Lepton module and actual temperature. This project does not require extreme precision, which is why the mapping function was empirically approximated using a linear function.

```
19      # Approximated using linear mapping function
20      @staticmethod
21      def Pixel2Temp(pixel, ambient_temp=22.0):
22          return 0.016632 * pixel - 50.93347 + ambient_temp
```

Illustration 7. Python function converting pixel flux value to an approximated temperature

4.2 Image processing

Image processing techniques play a very important role in the detection method we propose. Most of them are based on the OpenCV library (python ported)[7] as it is a very powerful tool in the area of image processing alongside the numpy library for python.

Since we already have an image with raw flux values and temperature mapping function, we can convert pixels into temperature values and create a binary mask for all pixels with a value in human temperature range using the numpy.inRange function. Before we apply the binary mask on the captured image, we need to provide values for areas outside the human

6 Thermometer datasheet https://www.sparkfun.com/datasheets/Sensors/Temperature/MLX90614_rev001.pdf

7 Python binding for the OpenCV library https://pypi.python.org/pypi/opencv-python

temperature range. Zero value cannot be used because this would negatively affect normalization process later on. Using the inverted binary mask we fill these areas with the higher value of lower range limit and global minimum pixel value. Now we can merge the original image with temperature mask applied and the second image with minimal values in areas outside the temperature range. In summary we have replaced all areas of the image with values outside the specified human temperature range by the lowest reasonable value so that we do not decrease dynamic range of the image.

Since this method is using primarily object or shape detection algorithm it is reasonable to perform normalization. We find global minimum and maximum pixel value in the image and map the values linearly between lowest and highest possible values. In fact, we are also lowering the resolution from 14 to 8 bits. In the next step we utilize OpenCV functionality and we perform adaptive binary Gaussian thresholding. Cv2.adaptiveThreshold takes two parameters and they were empirically set to 90 for box size and -30 for constant difference. As a result, we get a binary image (black and white) displaying white compact shapes of objects in human body temperature range on a black background of temperature of the environment. See top left figure of illustration 8.

In order to remove noise in the image, we use morphological transformations also available in OpenCV library. Firstly so called „opening" transformation was used which is erosion followed by dilation with kernel size 2 and secondly „closing" with kernel size 5 which is dilation followed by erosion. These transformations remove noise in the image and make all found shapes more consistent and compact.

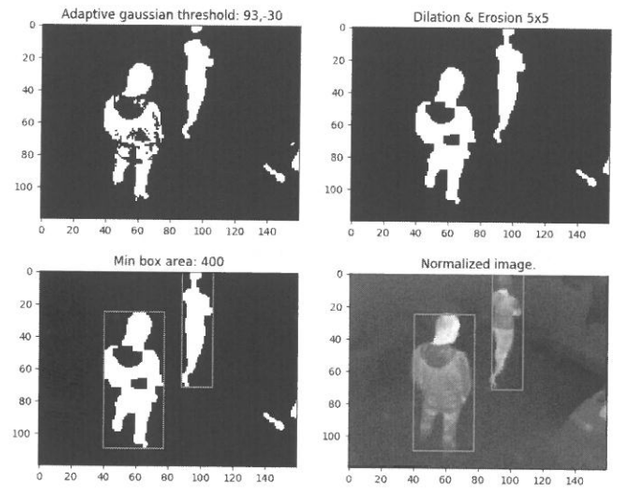


Illustration 8. Process of applying gaussian thresholding, filtering using morphological transformations and finding contours for border boxes

The core of the detection method lies in OpenCV's function findContours. The function returns a list of contours found in the image. These contours are edge points of compact shapes found in the image. It is recommended to apply this function only on binary image for best results. In order to differentiate between person and random small objects or noise, we calculate border box area for every contour and based on a set threshold value we can filter out small objects that are most likely not people. Furthermore, by using oriented area, we can discard black shapes on white background which would correspond to a cold object in front of a warm one (we only want warm objects on cold background). By acquiring a list of border boxes around detected persons we can naturally determine their count.

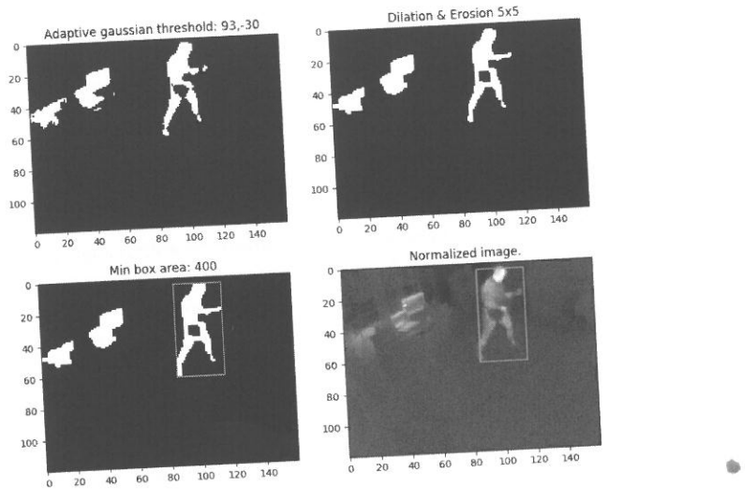


Illustration 9. Another example of the process of detecting people

4.3 Scene reconstruction

At this point we have bounding boxes around detected persons with image coordinates from a camera whose position is known in world space. Using the perspective-n-point algorithm, we are able to determine rotation and position of the camera in world space based on pairs: 3D world coordinate and 2D corresponding image coordinate. This functionality also comes with OpenCV and alongside the mapping points we also need matrix of intrinsic camera parameters. This matrix can be constructed with focal length in each axis and resolution. In our case, we calculate focal length fx, fy from horizontal and diagonal field of view, which is a known parameter of the Lepton 3. In this project we are not taking into count radial nor tangential distortion of the camera in this case. Cv2.solvePnP function returns rotation and translation vector, representing transformation from object to screen space (perspective projection). From rotation vector, we are able to acquire rotation matrix using Rodriguez algorithm and express the whole transformation from object to screen space using a single matrix.

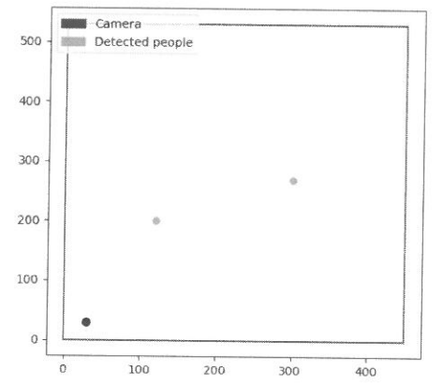


Illustration 10. Reconstructed scene from top view. Dark dot represents a camera, brighter dots detected people

Then, if we invert the matrix and apply a reverse transformation from screen space to object space, we are able to cast rays from the camera origin to the observed room intersecting position of feet or head of a detected person. We can then assume a person is standing on the floor or we can calculate with an average person height to collapse the ray into a single point in world space. This way, we are able to approximate person location in the observed space.

5. Conclusion

Even though the project should have been primarily focused on technique, different approaches regarding image processing and ways of detecting people, a big part of the project had to deal with the necessary hardware part – creating a working interface for the camera module and provide capture and control mechanisms. This part has been successfully accomplished. We have fully functional high level python script for controlling the Lepton 3 camera module, which can also be easily extended to support more commands thanks to the provided infrastructure. A single frame capturing python script along with implemented v4l2lepton3 C++ library opens various options for image processing. We can access frames from the camera directly using python or if we need sequential capture we can use faster option – using C++ library that feeds frames into a virtual video device from where one can process video frames using standard Unix tools or stream them over a network.

However, there is undoubtedly a lot of space and opportunities to improve and extend current relatively simple solutions of the detection algorithm. The method used in this project and described in the paper works perfectly in over 90 % of the cases on our data set. On one hand it is common that simpler solutions can sometimes work better and can be more reliable than complex solution. However, in this case, I believe that including other techniques might improve its accuracy even more – especially implementing motion tracking would increase the success rate and lower the probability of false positives. Then there are other approaches

for object detecting. One of them – so called YOLO (You Only Look Once) – is a technique based on a convolutional neural network and it is the current state-of-the-art in real-time object detection. This would be the ultimate solution to the problem of people detection, however it requires a big data set to be trained on and will not be as effective as current solution for unusual poses. This approach shall be a subject for further research in this topic.

6. Acknowledgement

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602 and the internal grant „Secure and Reliable Computer Systems“ FIT-S-17-4014.

7. References

Armbian (2017). Device Tree Overlays [online] https://docs.armbian.com/User-Guide_Allwinner_ovelays/

Armbian Forum (2018, April). Armbian on pcduino3 [online] https://forum.armbian.com/topic/382-armbian-on-pcduino3/#entry2444

DeMenthon, Daniel F. and Davis, Larry S. (2018, March). Model-Based Object Pose in 25 Lines of Code. University of Maryland [online] http://legacydirs.umiacs.umd.edu/~daniel/daniel_papersfordownload/Pose25Lines.pdf

FLIR Commercial Systems (2014). FLIR LEPTON3 Long Wave Infrared (LWIR) Datasheet. [online] https://www.flir.com/globalassets/imported-assets/document/lepton-3-engineering-datasheet-1.pdf

FLIR Commercial Systems (2014). FLIR LEPTON Software Interface Description Document. [online] https://www.flir.com/globalassets/imported-assets/document/flir-lepton-software-interface-description-document.pdf

FLIR Commercial Systems (2014). FLIR LEPTON Lepton vs. Lepton 3 Application Note. [online] https://www.flir.com/globalassets/imported-assets/document/lepton-vs-lepton-3-app-note.pdf

GroupGets (2018, April). Official software solutions for FLIR cameras [online] https://github.com/groupgets

GroupGets (2018, April). Pylepton. [online] https://github.com/groupgets/pylepton

GroupGets (2018, April). Quick start guide for Raspberry Pi 2 and FLIR Lepton. [online] https://files.groupgets.com/lepton/Raspberry_Pi_-_%20Lepton_Quickstart_Guide-v3.pdf

GroupGets (2018, April) v4l2lepton repository. [online] https://github.com/groupgets/LeptonModule/tree/master/software/v4l2lepton

Mallick, S. (2016, September). Head Pose Estimation using OpenCV and Dlib. [online] https://www.learnopencv.com/head-pose-estimation-using-opencv-and-dlib/

Mordvintsev, A. & Abid, K. (2013). OpenCV-Python Tutorials's documentation. [online] http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html

OpenCV Documentation (2017, August). Pose estimation. [online] https://docs.opencv.org/3.3.0/d7/d53/tutorial_py_pose.html

Pahhomov, D. (2016, September) Install OpenCV and Python on Orange Pi PC Plus SBC. LYNX [online] https://lynx2015.wordpress.com/2016/09/20/install-opencv-and-python-on-orange-pi-pc-plus-sbc/

Raval, S. (2017, November). YOLO Object Detection. [online] https://youtu.be/4eIBisqx9_g

Solem, J. E. (2012, June). Programming Computer Vision with Python. O'Reilly Media, Inc., Chapter 4 Camera Models and Augmented Reality. [online] https://www.safaribooksonline.com/library/view/programming-computer-vision/9781449341916/ch04.html

Vegard (2015, November). How do I reverse-project 2D points into 3D [online] https://stackoverflow.com/questions/76134/how-do-i-reverse-project-2d-points-into-3d

Xunlong (2018, April) Xunlong Orange Pi PC 2. [online] http://linux-sunxi.org/Xunlong_Orange_Pi_PC_2