DOLPi – A Low-Cost, RasPi-based Polarization Camera Duo

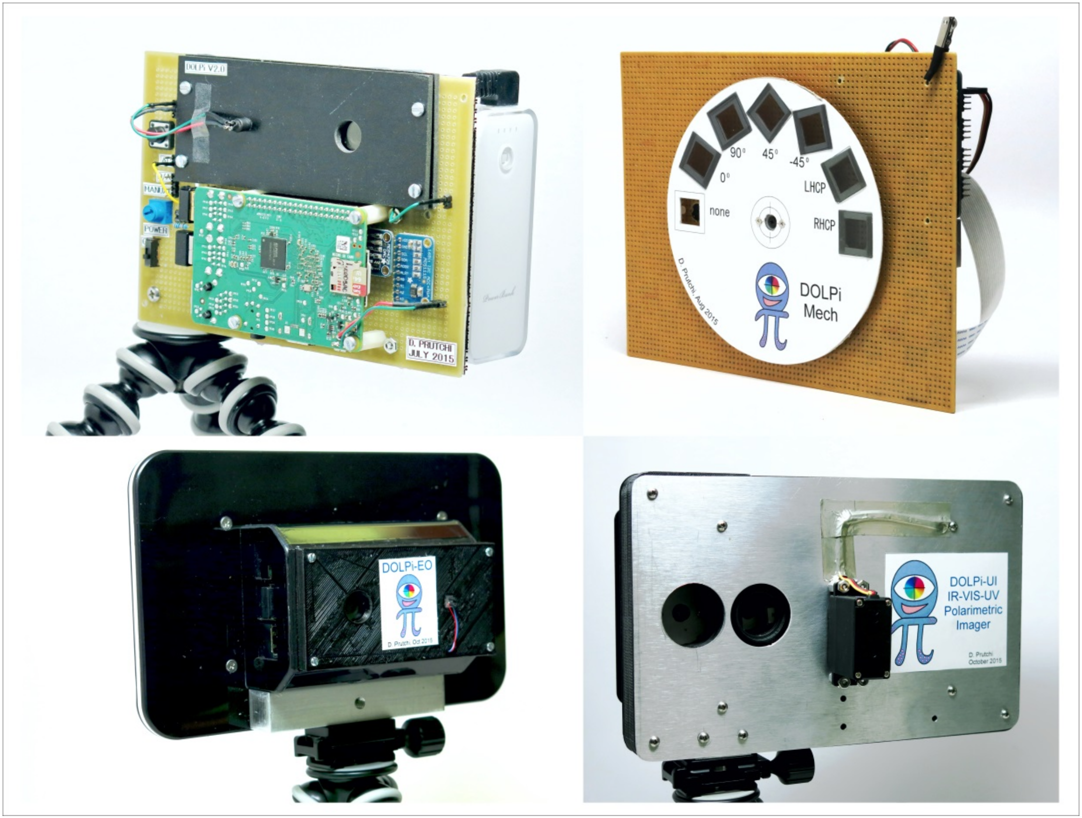
Two easy-to-build polarimetric imagers to locate landmines and IEDs, detect invisible pollutants, identify cancerous tissues, and maybe even observe cloaked UFOs.

David Prutchi, Ph.D.

www.diyPhysics.com

Summary

This paper presents the development and construction of two low-cost polarimetric camera types based on the Raspberry Pi 2. DOLPi-MECH (and its productized IR-VIS-UV version DOLPi-UI) is a filter-wheel-type camera capable of performing full Stokes analysis, while the electro-optic based DOLPi-EO camera performs full linear polarimetric analysis at higher frame rate. Complete Python code for polarimetric imaging is presented. Various applications for the cameras are described, especially their use for locating mines and unexploded ordinance in humanitarian demining operations.



Introduction

"If you make a new instrument and point it at nature you will learn something new."

- *D. Staelin*[1]

Clouds of colorless pollutants, antipersonnel mines, and skin cancers are so difficult to detect because they blend in so naturally with their background that our unaided eyes cannot see them[2]. However, an octopus faced with the same problem would probably have a very easy time at locating these threats. This is because our human eyes can only distinguish objects from their background through the contrast in their color and/or intensity (Figure 1). On the other hand, octopuses would be able to see the contrast in light **polarization** between these items and their background.

Try this experiment - next time that you go outdoors on a sunny day, tilt your head while wearing polarized sunglasses. You will find that parts of the sky turn into a lighter shade of blue when you turn your head sideways. The intensity of the glare will also change considerably as you tilt your head while looking at a reflective surface like still water, or the windshield of a car. Look at a static scene like a parking lot while tilting your head back and forth – do the windows of parked cars seem to flash? The reason for this phenomenon is that the light scattered by the sky, or reflected by many surfaces is "polarized". That is to say that light waves from these sources vibrate mainly in one direction (Figure 1) – more on this later.

In fact, every photon that reaches our eyes has its own polarization, yet this aspect of light is barely used in our daily life because our unaided eyes are insensitive to polarization[3], and thus we don't have an intuitive sense for its use.

In spite of this, polarization of light carries interesting information about our visual environment of which we are usually unaware. Some animals have evolved the capability to see polarization as a distinct characteristic of light, and rely critically on this sense for navigation and survival. For example, many fish, amphibians, arthropods, and octopuses[4] use polarization vision as a compass for navigation, to detect water surfaces, to enhance the detection of prey and predators, and probably also as a private means to communicate among each other.

[1] Late MIT Prof. David Staelin as quoted by fellow hardware hacker Gregory L. Charvat.
[2] You'll have to keep on reading to find the reference about observing cloaked UFOs.
[3] Humans have very marginal sensitivity to polarized light as discovered by Haidinger in 1846, but changes in polarization can only be perceived under very specific conditions and do not contribute to visual feature discrimination.
[4] "Octopuses" is the correct plural of "octopus."

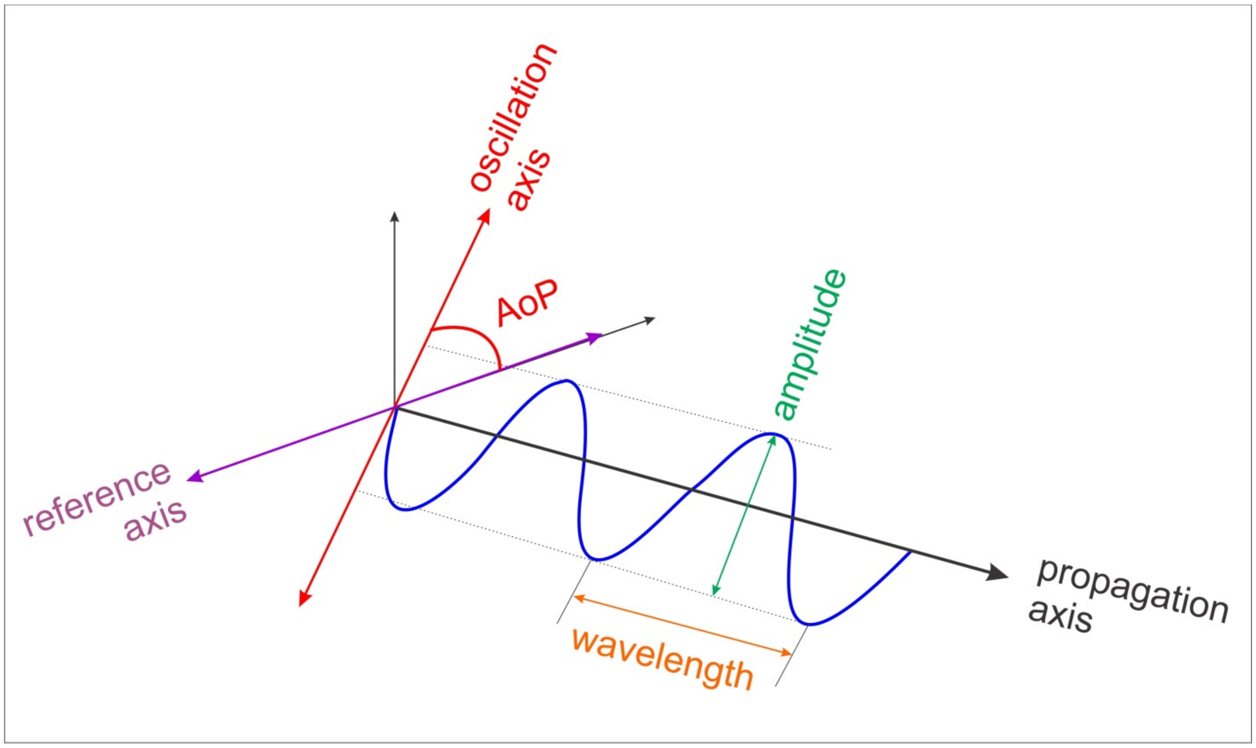


Figure 1 – A light wave is characterized by its wavelength, which we perceive as a district color; its amplitude, which we perceive as an intensity level; and the angle at which it oscillates with respect to a reference axis. This last parameter is called the wave's "Angle of Polarization", and is a characteristic of light that unaided human eyes cannot distinguish. However, the polarization of light carries interesting information about our visual environment, and some animals are able to perceive it and rely critically on this sense for navigation and survival.

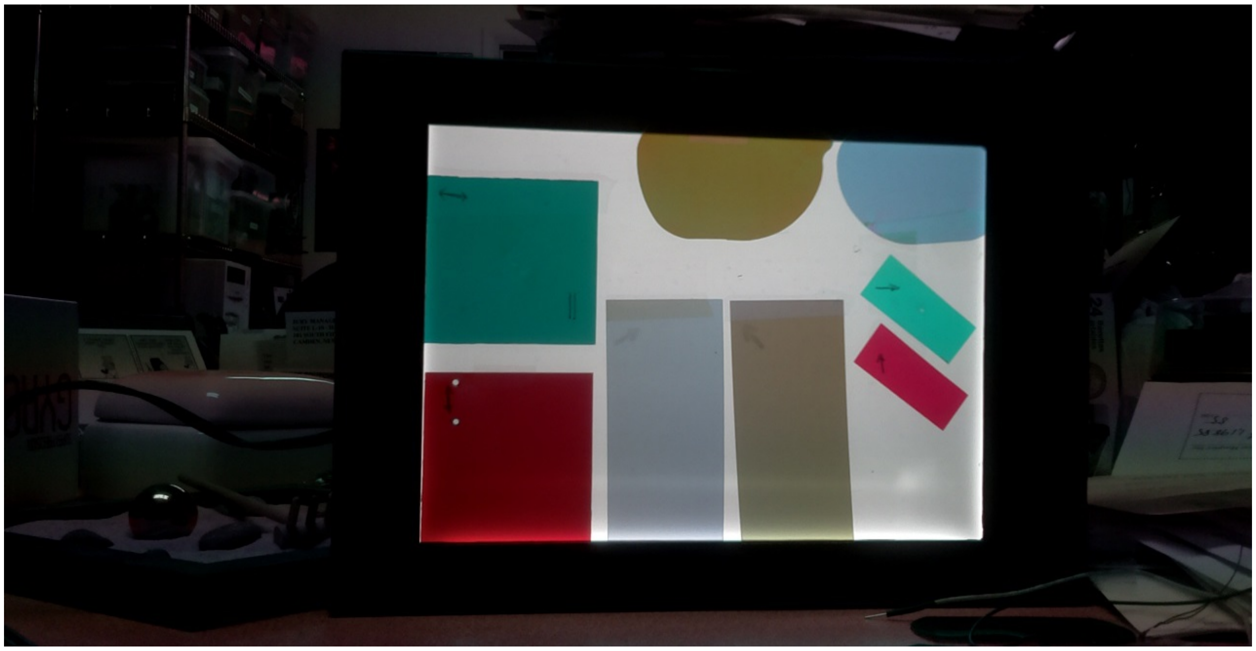


Figure 2 - In spite of its simplicity and low cost, DOLPi camera produces very high quality images. Here is a screen capture of DOLPi viewing a few pieces of back-illuminated polarizing film. Unpolarized light shows as grayscale, while only polarized light is given a color representative of the angle of polarization.

While we have used technology to expand our vision beyond the limits of our ordinary wavelength and intensity sensitivities, the unintuitive nature of polarization has slowed down the development of practical applications for polarization imaging. Polarization cameras do exist, but at over $20,000, they are mostly research curiosities that have found very few practical uses outside the lab.

The DOLPi project[5] aims to widely open the field of polarization imaging by constructing a very low cost polarization camera that can be used to research and develop game-changing applications across a wide range of fields – spanning all the way from environmental monitoring and medical diagnostics to security and antiterrorism applications.

The DOLPi polarization camera is based on a standard Raspberry Pi 2 single-board computer and its dedicated 5MP camera. What makes the DOLPi unique is that the camera sits behind a software-controlled electro-optic polarization modulator, allowing the capture of images through an electronic polarization analyzer. The modulator itself is hacked from a low-cost auto-darkening welding mask filter ($9 on eBay®). In spite of its simplicity, DOLPi produces very high quality polarization images such as the one shown in Figure 2. The construction of a slower, but more accurate filter-wheel-based polarimetric camera (DOLPi-Mech), as well as an all-mode imager (DOLPi-UI) capable of infrared-visual-ultraviolet imaging and polarimetry are also presented.

This is a first-of-its-kind project! I am not aware of any polarization imager ever presented as an enthusiast-level DIY project, yet it holds truly awesome disruptive power for the development of brand new scientific and commercial applications!

[5] Although DoLP refers only to the Degree of Linear Polarization, I liked the name DOLPi as a combination of DOLP and Pi (for Raspberry Pi). The logo is a one-eyed octopus (squids have polarization-sensitive eyes) with "Pi" as its tentacles.

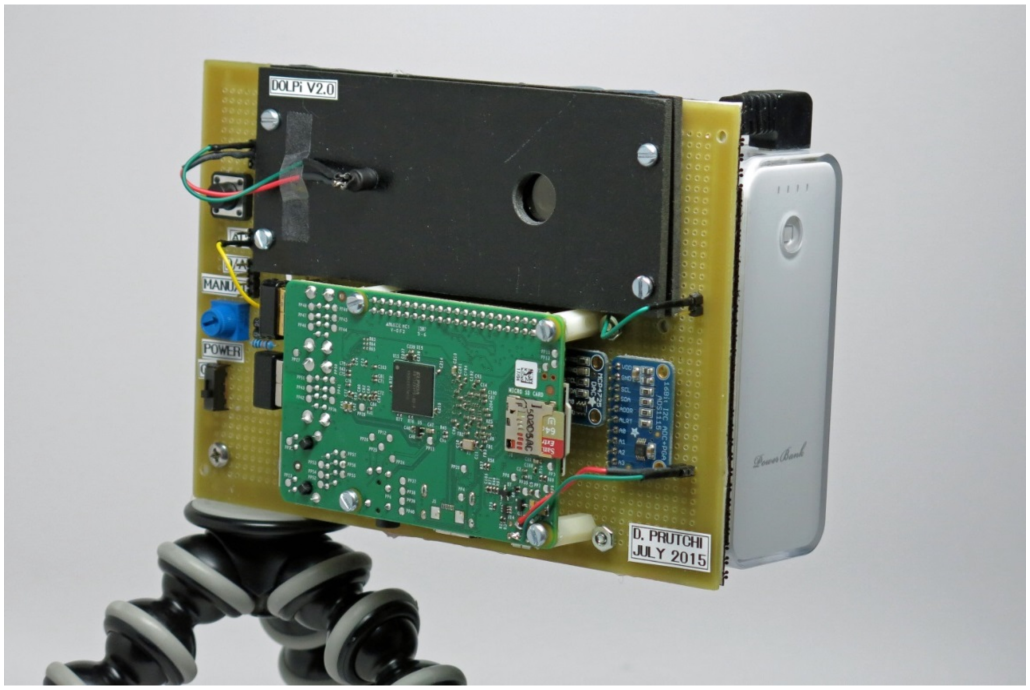


Figure 3 – The DOLPi camera is fully self-contained so it can be easily taken wherever needed to perform high-quality polarization imaging, rendering not only the images corresponding to the linear Stokes parameters, but also to Polarization Intensity, Degree of Linear Polarization, Angle of Polarization, and their HSV rendering. The productized version of this camera (Figure 4) is described in Appendix V.

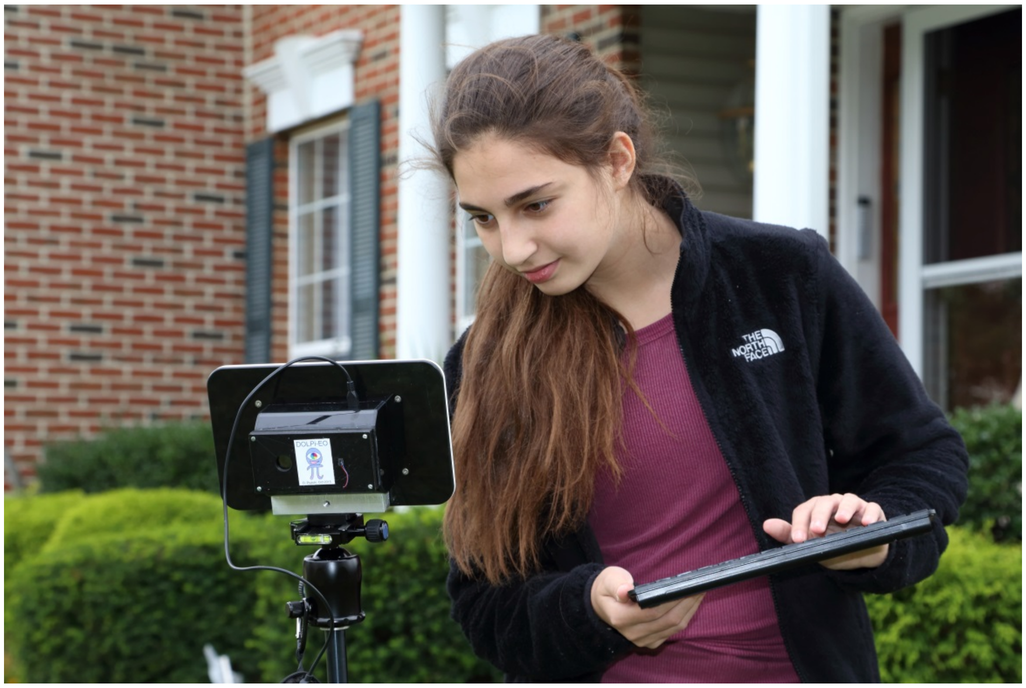


Figure 4 – The productized version of the electro-optical DOLPi polarimetric camera is self-contained and incorporates RPi Foundation's official 7" touchscreen display into a single enclosure. Shown in this photograph is my daughter Abigail helping me run a simulated landmine detection test.

Polarization and Polarizers

Polarization is an important characteristic of light that stems from its nature as an electromagnetic wave. In 1860, Scottish physicist James Clerk Maxwell figured out that an electric field that varies along space generates a magnetic field that varies in time and vice versa. For that reason, as an oscillating electric field generates an oscillating magnetic field, the magnetic field in turn generates an oscillating electric field, and so on. These oscillating fields together form the electromagnetic wave shown in Figure 5. In a wave of polarized light, like the one shown in the figure, the electric field oscillates in one plane, while the magnetic field oscillates on a perpendicular plane. The wave travels at the speed of light along the line formed by the intersection of those planes. The electromagnetic wave shown in this figure is said to be "vertically polarized" because the electric field oscillates vertically in the frame of reference that we have chosen.

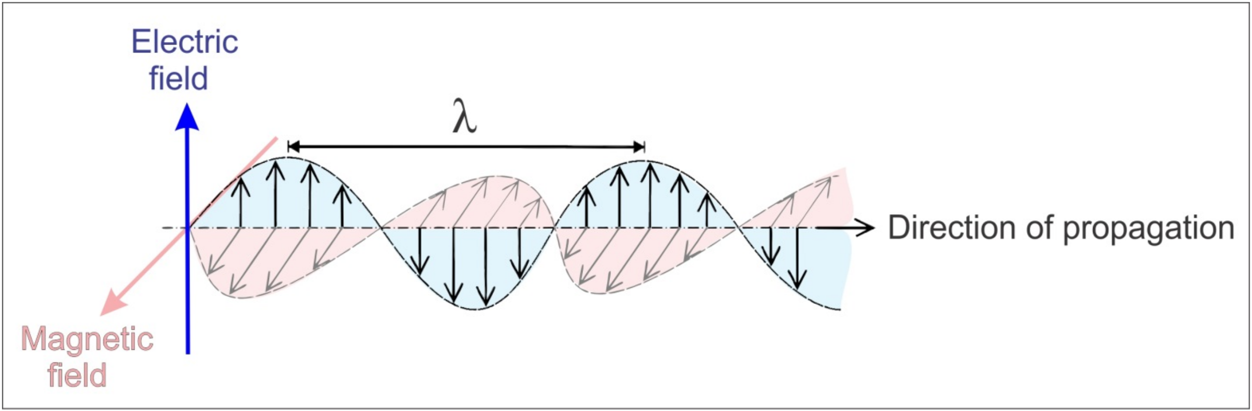


Figure 5 - An oscillating electric field generates an oscillating magnetic field, the magnetic field in turn generates an oscillating electric field, and so on. These oscillating fields together form an electromagnetic wave with wavelength λ that propagates at the speed of light. Adapted with permission from D. Prutchi and S.R. Prutchi, *Exploring Quantum Physics through Hands-On Projects*, John Wiley & Sons, Inc., 2012.

Light from most natural sources contains waves with electric fields oriented at random angles around its direction of travel. A wave of a specific polarization can be obtained from randomly-polarized light by using a polarizer filter. As shown in Figure 6, a polarizer can be made of an array of very fine wires arranged parallel to one another. The metal wires offer high conductivity for electric fields parallel to the wires, essentially "shorting them out" and producing heat. Because of the nonconducting spaces between the wires, no current can flow perpendicularly to them. As such, electric fields perpendicular to the wires can pass unimpeded. In other words, the wire grid, when placed in a randomly-polarized beam, drains the energy out of one component of the electric field and lets its perpendicular component pass with no attenuation at all. Thus, the light emerging from the polarizer has an electric field that vibrates in a direction perpendicular to the wires.

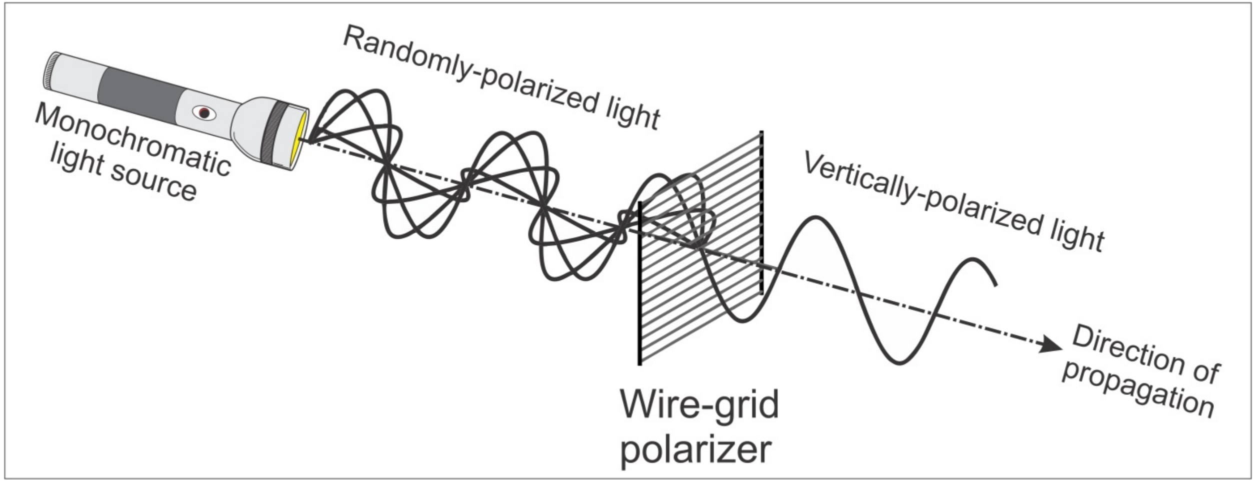


Figure 6 - A parallel-wire polarizer absorbs electric-field lines that are parallel to the wires. Only the perpendicular electrical field component of light is allowed to pass, producing light that is polarized perpendicularly to the direction of the wires. Adapted with permission from D. Prutchi and S.R. Prutchi, *Exploring Quantum Physics through Hands-On Projects*, John Wiley & Sons, Inc., 2012.

Although the wire-grid polarizer is easy to understand, it is useful only down to certain wavelengths because the wires have to be a fraction of the wavelength apart. This is difficult and expensive to do for short wavelengths such as those of visible light[6]. In 1938 E. H. Land invented the H-Polaroid sheet, which acts as a chemical version of the wire grid. Instead of long thin wires it uses long thin polyvinyl alcohol molecules that contain many iodine atoms. These long, straight molecules are aligned almost perfectly parallel to one another. Because of the conductivity provided by the iodine atoms, the electric vibration component parallel to the molecules is absorbed. The component perpendicular to the molecules passes on through with little absorption.

[6] Wire-grid polarizers can now be made with lithography techniques and are used as part of micropolarizer arrays. See Figure 32-c. In addition, new demand for high-performance filters has driven the development of relatively low cost films, which although not as cheap as Polaroid, are nevertheless affordable. I used this type of wire-grid polarizing film in the construction of DOLPi-UI as described in Appendix IV.

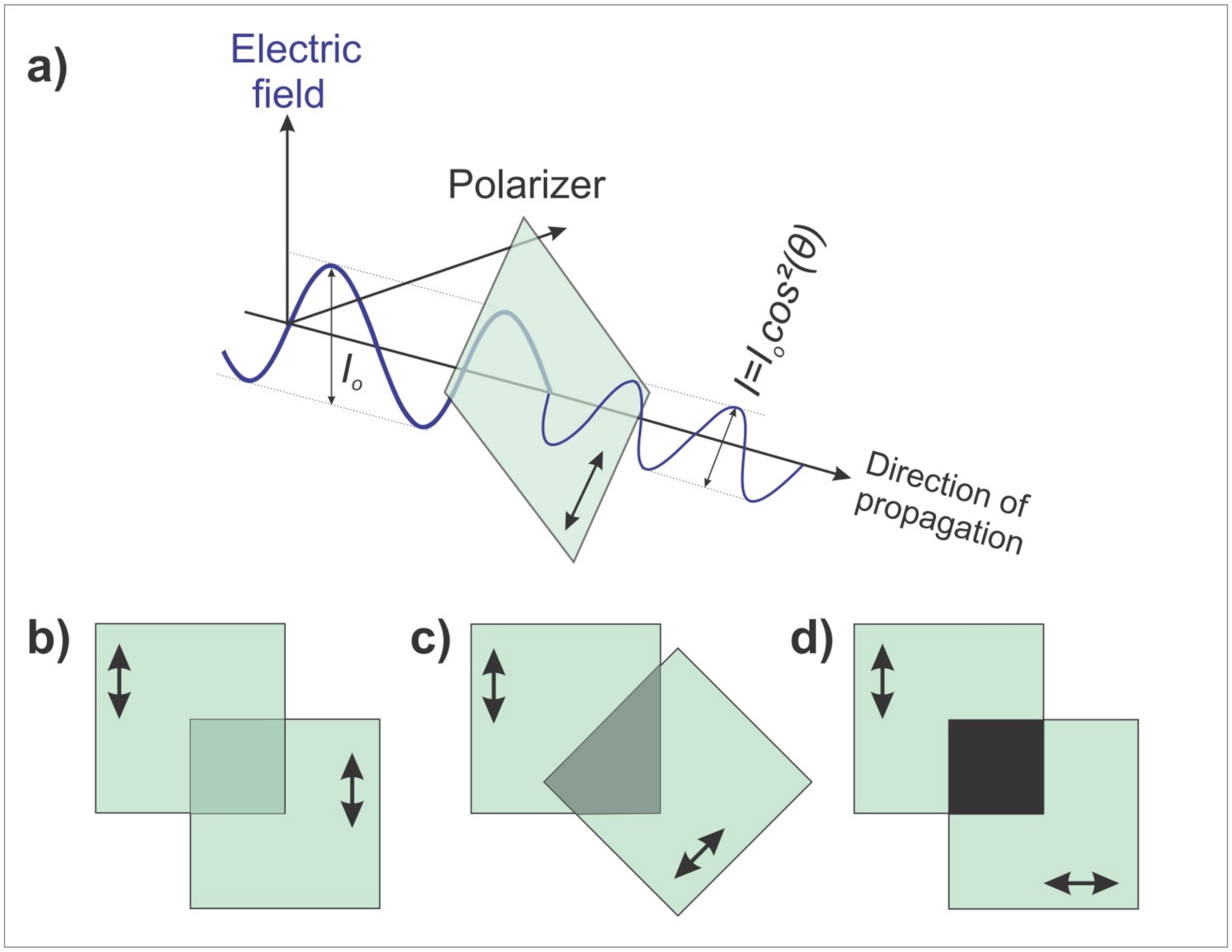


Figure 7 – The amount of polarized light that passes through a polarizer depends on the angle between the polarized light's axis and the polarizer's main axis. a) The intensity of the light exiting the polarizer follows the cosine square of the angle between the light's polarization axis and the polarizer's main axis. With light going through two pieces of polarizer, minimum attenuation happens when the axes of polarization of the pieces of film are aligned (b). Attenuation increases as the angle between the axes increases (c) until a maximum attenuation occurs when the films are cross-polarized.

The amount of polarized light that passes through a polarizer depends on the angle between the polarized light's axis and the polarizer's main axis. As shown in Figure 7, the intensity of the light exiting the polarizer follows the cosine square of the angle between the light's polarization axis and the polarizer's main axis. Two polarizers in series will attenuate non-polarized light by an amount dependent on the rotation between the polarizers. Maximum transmission will happen when the polarizers are aligned ($\cos^2(0°)=1$), while minimum transmission will take place with the polarizer orientations crossed ($\cos^2(90°)=0$). Half of the maximum intensity is observed at a rotation angle of 45º.

Besides linear polarization, light can also be circularly polarized. This type of polarization is obtained when light polarized at 45º passes through a material that transmits light at different speeds depending on its polarity. Linear polarization is thus given a "twist" when one polarization component (e.g. the vertical component of the 45º light) is retarded by ¼ of its wavelength with respect to the other component (e.g. the horizontal component). When exiting the retarder material, the vertical component is slowed down so that it is out of phase with the horizontal component. To an observer receiving this light, the electric field will appear to rotate rather than just oscillate up and down. Hence the term “circular” polarization. As shown in Figure 8, circularly-polarized light can be polarized either clockwise (right-handed circular polarization or RHCP) or counterclockwise (left-handed circular polarization or LHCP).

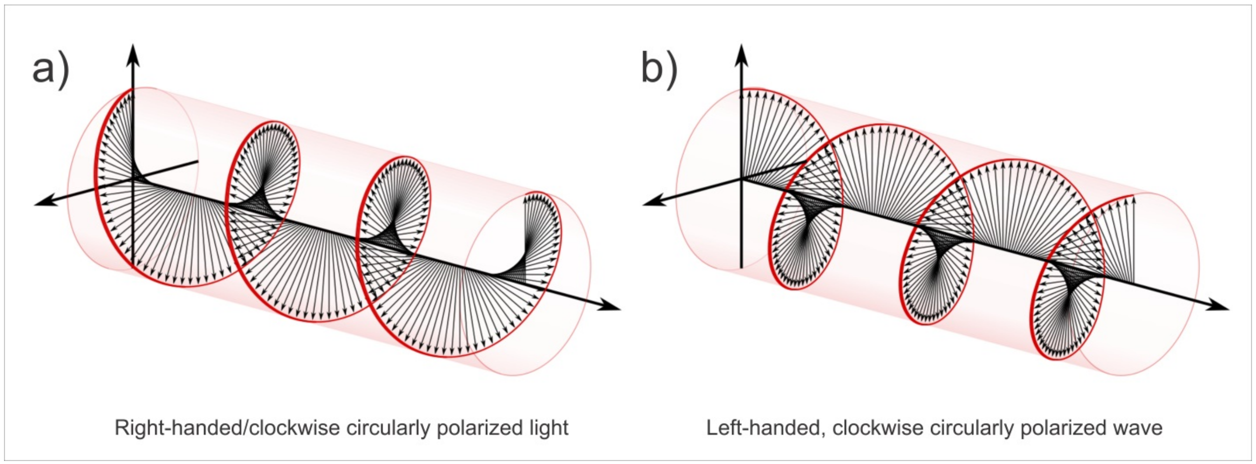

a)
b)
Right-handed/clockwise circularly polarized light
Left-handed, clockwise circularly polarized wave

Figure 8 – Circularly-polarized light. The convention is to define the sense of polarization as viewed by the receiver, so (a) represents the electric field vector as a function of time of right-handed, clockwise circularly polarized light, while (b) represents left-handed, clockwise circularly polarized light. Images used in figure by anonymous via Wikimedia Commons.

Circularly-polarized light is easy to produce, and is used extensively in 3D movie projection (e.g. the RealD digital stereoscopic projection system encodes one image in RHCP, and the other in LHCP). However, natural materials that circularly-polarize light are very scarce, so restricting the analysis of a scene to its linear polarization characteristics is appropriate for most applications.

DOLPI-Mech: A RasPi-based Classical Polarimetric Camera

Further down this paper we’ll look at the mathematical formalism behind the analysis of polarized light. For now, let me state that complete linear polarization state analysis can be performed on a scene from 3 individual images taken through a linear polarizer set at 0^{o}, 45^{o}, and 90^{o}. In addition, the handedness of circular polarization can be obtained if an additional picture is obtained through a circular polarizer.

One simple way of obtaining the necessary images is to use a filter wheel. That is, a mechanically-rotated wheel that carries different filters in front of a camera. Figure 9 shows the construction of DOLPi-Mech - a simple Raspberry Pi-based polarimetric imager with a mechanically-rotated cardboard filter wheel that holds 6 polarizer filters (Figure 10). Four of them are made of linear polarizer film set so to analyze the image at 0^{o}, 90^{o}, 45^{o}, and -45^{o} when the selected filter is placed in front of the Raspberry Pi camera. The other two filters are circular polarizer films. One slot in the filter wheel is left blank to make it possible to take unfiltered snapshots. The filter wheel is rotated by a standard servo driven by an Adafruit Servo PWM HAT connected to a Raspberry Pi 2.

The Servo PWM hat is not absolutely necessary. The servo could be driven directly by the Raspberry Pi getting the Pi Linux kernel to generate the PWM signal. However, the lack of precision makes the servo jittery and consumes processing time, so I chose to use the dedicated hardware solution.

The Python listing for this simple polarization camera is shown in Table 2. Figure 11 shows a simplified flowchart for the software. The program first imports necessary packages and then allows the camera to sample the scene before locking its exposure settings. The imaging loop consists of taking three grayscale images with the filter wheel placing 0^o, 45^o, and 90^o polarization filters in sequence, and then combining these pictures into a single color image encoding the scene's intensity and polarization.

Table 1 – Bill of Materials for DOLPi-Mech (some may be replaced or are optional)

Component reference	Component/Material	Source
RasPi	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	Adafruit ID:2358
SD Card	SanDisk SAEMSD64GBU3 64GB Extreme UHS-I microSDXC Memory Card (U3/Class 10)	B&H Photo
+5V Power Supply	5V 2A Switching Power Supply w/ 20AWG 6' MicroUSB Cable	Adafruit ID: 1995
RasPi Camera	Raspberry Pi Camera Board	Adafruit ID:1367
HDMI Display	HDMI Display	Any suitable HDMI display
Keyboard/Mouse	USB keyboard and mouse	Any suitable keyboard and mouse
Servo Controller	Adafruit 16-Channel PWM/Servo HAT for Raspberry Pi	Adafruit ID: 2327
Servo	Standard $\pm 90^o$ hobby servo	Adafruit ID: 155
Linear Polarizer Film	Polarizing Film Sheet - set of 10	Amazon.com
Circular Polarizer Film	RHCP and LHCP polarizer films taken from RealD 3D movie glasses	RealD movie glasses
Perfboard	6" x 8" perfboard	Radio Shack
Misc. hardware	Nylon standoffs, bolts/nuts, Delrin block for tripod mount, black plastic board, industrial-grade hook&loop tape, etc.	

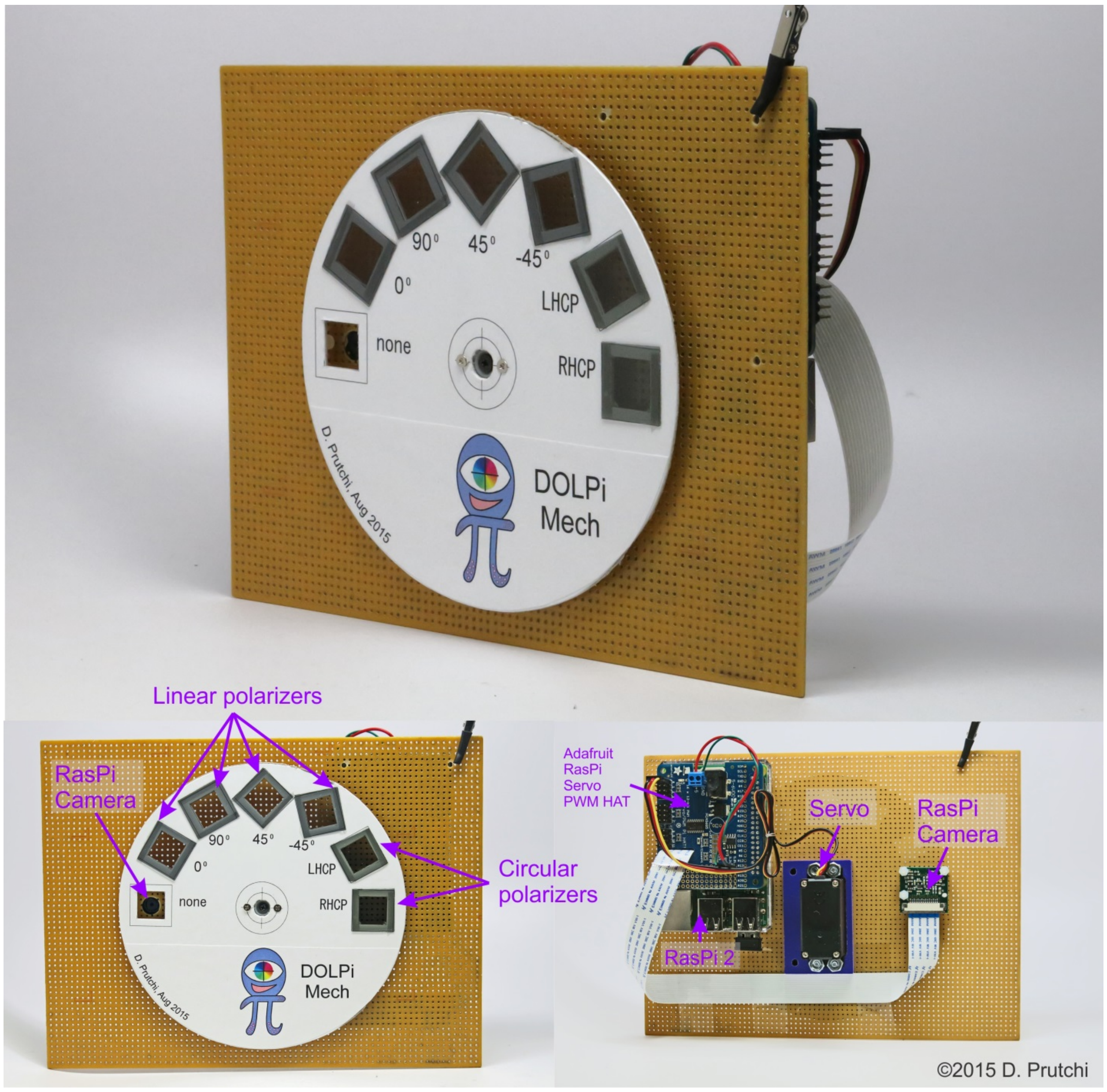


Figure 9 – DOLPi-Mech is a simple polarimetric imager based on a mechanically-rotated cardboard filter wheel that holds 6 polarizer filters. Four of them are made of linear polarizer film set so to analyze the image at 0°, 90°, 45°, and -45° when the selected filter is placed in front of the Raspberry Pi camera. The other two filters are circular polarizer films. One slot in the filter wheel is left blank to make it possible to take unfiltered snapshots. The filter wheel is rotated by a standard-size 180° servo driven by an Adafruit Servo PWM HAT connected to a RasPi2. Please also look at the more capable DOLPi-UI all-mode imager presented in Appendix IV.

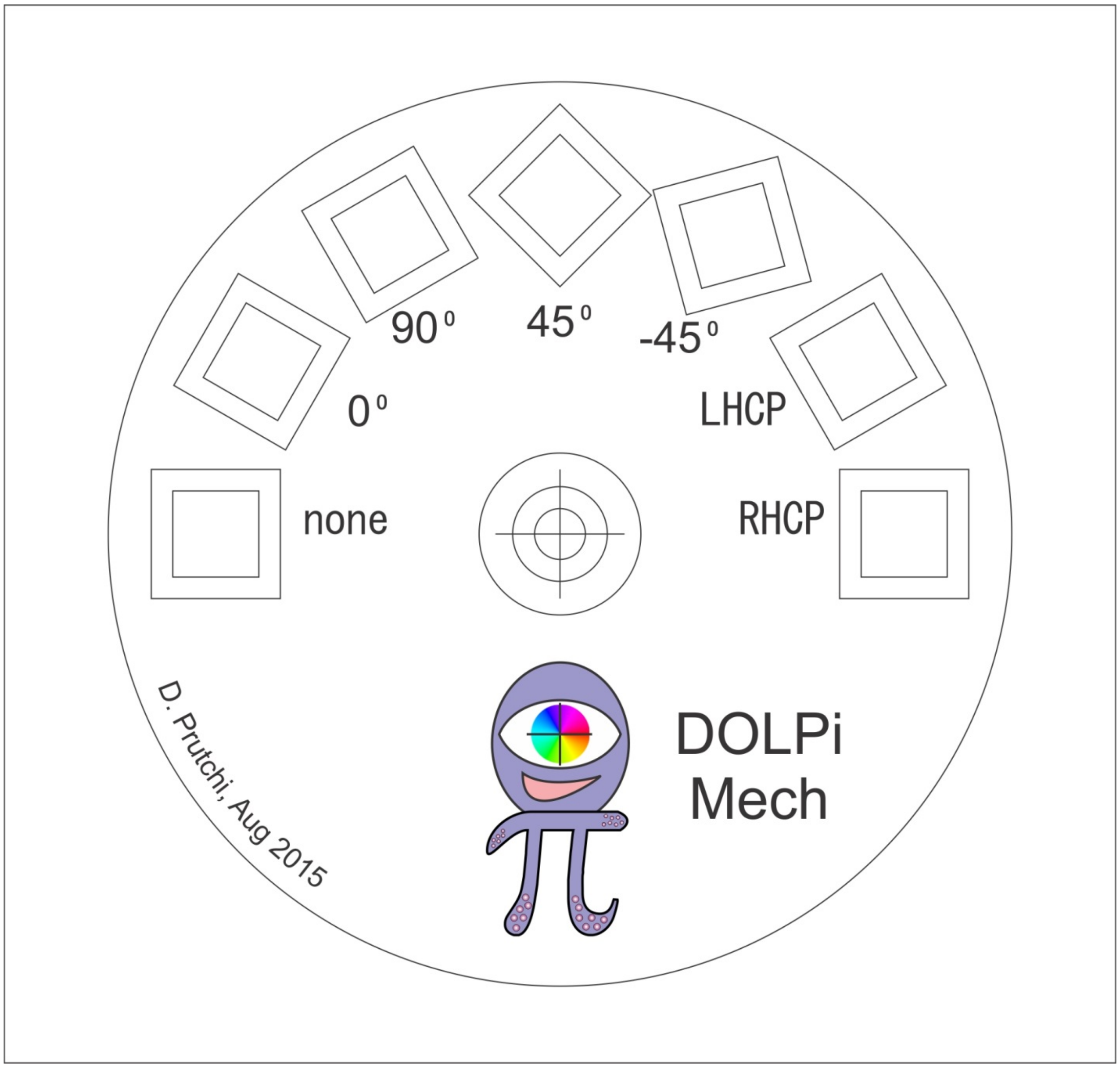


Figure 10 – Template for the DOLPi-MECH filter wheel. The diameter of the wheel is 5.25”. It can be printed on paper and glued to light cardboard. The center circles are used to align and mount the filter wheel to a standard circular servo hub. Pieces of polarizer film are adhered in place using thin strips of clear double-sided tape.

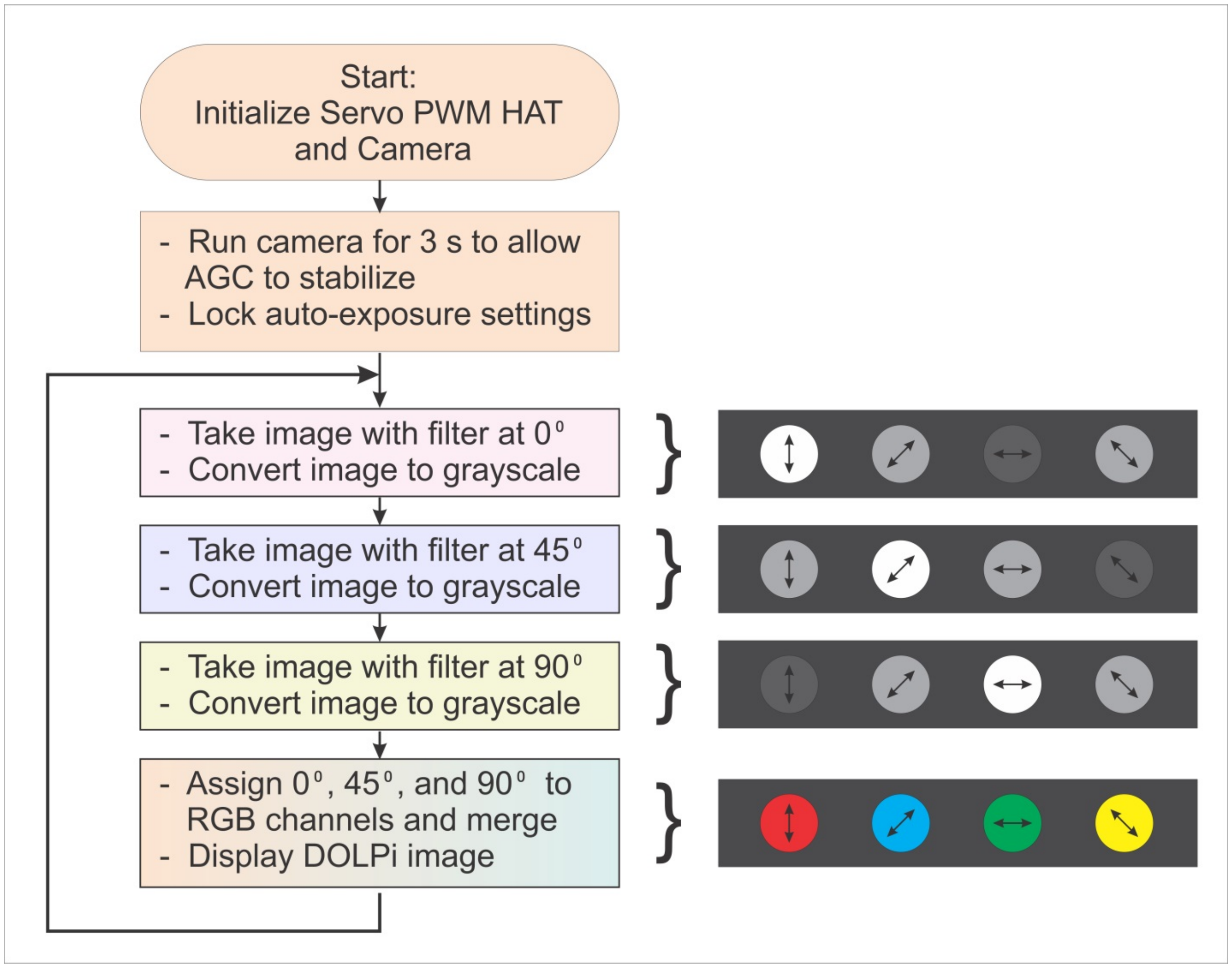

Start:
Initialize Servo PWM HAT and Camera
- Run camera for 3 s to allow AGC to stabilize
- Lock auto-exposure settings
- Take image with filter at 0°
- Convert image to grayscale
- Take image with filter at 45°
- Convert image to grayscale
- Take image with filter at 90°
- Convert image to grayscale
- Assign 0°, 45°, and 90° to RGB channels and merge
- Display DOLPi image

Figure 11 - This flowchart explains the simple software used to image with the camera of Figure 9. After loading libraries and initializing the Raspberry Pi's ports, the Raspberry camera is allowed to sample the scene for a few seconds to set its internal auto-exposure and then locking the exposure settings to keep consistency among images. The imaging loop consists of taking three grayscale images with the filter wheel selecting linear polarization filters set at 0°, 45°, and 90°, and then combining these pictures into a single color image that encodes polarization.

Table 2 – DOLPiMech.py Python code

```
#   DOLPiMech.py
#
#   This Python program demonstrates the DOLPi_Mech polarimetric camera.
#
#   A servo motor rotates a polarization filter wheel in front of the
#   Raspberry Pi camera.  An Adafruit PWM Servo HAT drives the servo.
#
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                          (MIT, opensource.org/licenses/MIT)
#
#
#import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
from Adafruit_PWM_Servo_Driver import PWM
import numpy as np

# Initialise the Adafruit PWM HAT using the default address
pwm = PWM(0x40)
pwm.setPWMFreq(60)   # Set PWM frequency to 60 Hz
servoMin = 150       # Min pulse length out of 4096
servoMax = 615       # Max pulse length out of 4096
servoNone = 615      # PWM setting for open window
servo0=540          # PWM setting for 0 degree filter
servo90=465         # PWM setting for 90 degree filter
servo45=390         # PWM setting for 45 degree filter

# PWM setting function
def setServoPulse(channel, pulse):
  pulseLength = 1000000                   # 1,000,000 us per second
  pulseLength /= 60                       # 60 Hz
  print "%d us per period" % pulseLength
  pulseLength /= 4096                     # 12 bits of resolution
  print "%d us per bit" % pulseLength
  pulse *= 1000
  pulse /= pulseLength
  pwm.setPWM(channel, 0, pulse)

# Rotate filter wheel back and forth to check all is OK
pwm.setPWM(0, 0, servoMin)
time.sleep(1)
pwm.setPWM(0, 0, servoMax)
time.sleep(1)

#Raspberry Pi Camera Initialization
#----------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
#camera.resolution = (1280,720)
camera.framerate=30
rawCapture = PiRGBArray(camera)
camera.led=False

#Auto-Exposure Lock
#------------------
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
```

```
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain

#Initialize flags
loop=True  #Initial state of loop flag
first=True #Flag to skip display during first loop
video=False #Use video port?  Video is faster, but image quality is significantly
        #lower than using still-image capture

while loop:
  #grab an image from the camera at 0 degrees
  pwm.setPWM(0, 0, servo0)
  time.sleep(0.1)   #Wait for filter wheel to moves
  rawCapture.truncate(0)
  camera.capture(rawCapture, format="bgr",use_video_port=video)
  image0 = rawCapture.array
  # Select one of the two methods of color to grayscale conversion:
  # Blue channel gives better polarization information because wavelengt
  # range is limited
  # True grayscale conversion gives better gray balance for non-polarized light
  #R=image0[:,:,1]  #Use blue channel
  R=cv2.cvtColor(image0,cv2.COLOR_BGR2GRAY) #True grayscale conversion

  #grab an image from the camera at 90 degrees
  pwm.setPWM(0, 0, servo90)
  time.sleep(0.1)
  rawCapture.truncate(0)
  camera.capture(rawCapture, format="bgr",use_video_port=video)
  image90 = rawCapture.array
  # Select one of the two methods of color to grayscale conversion:
  # Blue channel gives better polarization information because wavelengt
  # range is limited
  # True grayscale conversion gives better gray balance for non-polarized light
  #B=image90[:,:,1]  #Use blue channel
  B=cv2.cvtColor(image90,cv2.COLOR_BGR2GRAY) #True grayscale conversion

  #grab an image from the camera at 45 degrees
  pwm.setPWM(0, 0, servo45)
  time.sleep(0.1)
  rawCapture.truncate(0)
  camera.capture(rawCapture, format="bgr",use_video_port=video)
  image45 = rawCapture.array
  # Select one of the two methods of color to grayscale conversion:
  # Blue channel gives better polarization information because wavelengt
  # range is limited
```

```
    # True grayscale conversion gives better gray balance for non-polarized light
    #G=image45[:,:,1]  #Use blue channel
    G=cv2.cvtColor(image45,cv2.COLOR_BGR2GRAY) #True grayscale conversion

    imageDOLPi=cv2.merge([B,G,R])
    cv2.imshow("Image_DOLPi",imageDOLPi)  #Display DOLP image
    #cv2.imshow("Image_DOLPi",cv2.resize(imageDOLPi,(320,240),interpolation=cv2.INTER_AREA))
#Display DOLP image
    k = cv2.waitKey(1)  #Check keyboard for input
    if k == ord('x'):   # wait for x key to exit
        loop=False

#   Prepare to leave
#   ----------------
#
#  Take a "normal" picture
pwm.setPWM(0, 0, servoNone)
time.sleep(0.2)
rawCapture.truncate(0)
camera.capture(rawCapture, format="bgr",use_video_port=video)
imagenone = rawCapture.array
cv2.imwrite("imagenone.jpg",imagenone)
cv2.imwrite("image0.jpg",image0)
cv2.imwrite("image90.jpg",image90)
cv2.imwrite("image45.jpg",image45)
cv2.imwrite("RGBpol.jpg",cv2.merge([B,G,R]))
cv2.imwrite("image0g.jpg",R)
cv2.imwrite("image90g.jpg",G)
cv2.imwrite("image45g.jpg",B)
cv2.destroyAllWindows()
quit
```

The picture shown in Figure 12-b is a sample image obtained through DOLPi-Mech and the Python code of Table 2 (linear analysis only, using pictures acquired with analyzer set at 0^o, 90^o, and 45^o). For comparison, the picture of Figure 12-a is unfiltered (RasPi camera looking through open slot in filter wheel).

The sample target is made of polarizing film set at the angles shown in the picture. The two bottom squares are circular polarizers (one LHCP and the other RHCP, taken from RealD 3D movie glasses). We can’t tell the difference between the polarizer films because our vision is not sensitive to polarization, and neither is the RasPi’s camera sensor.

The colors in the picture of Figure 12-b have nothing to do with the actual color of the object, but instead encode the Angle of Polarization. Pure gray tones in the polarimetric image mean that light is unpolarized. Some of the coloring pencils and other items in the background show in color because they partially polarize reflected light as a function of their material, texture, and angle of reflection.

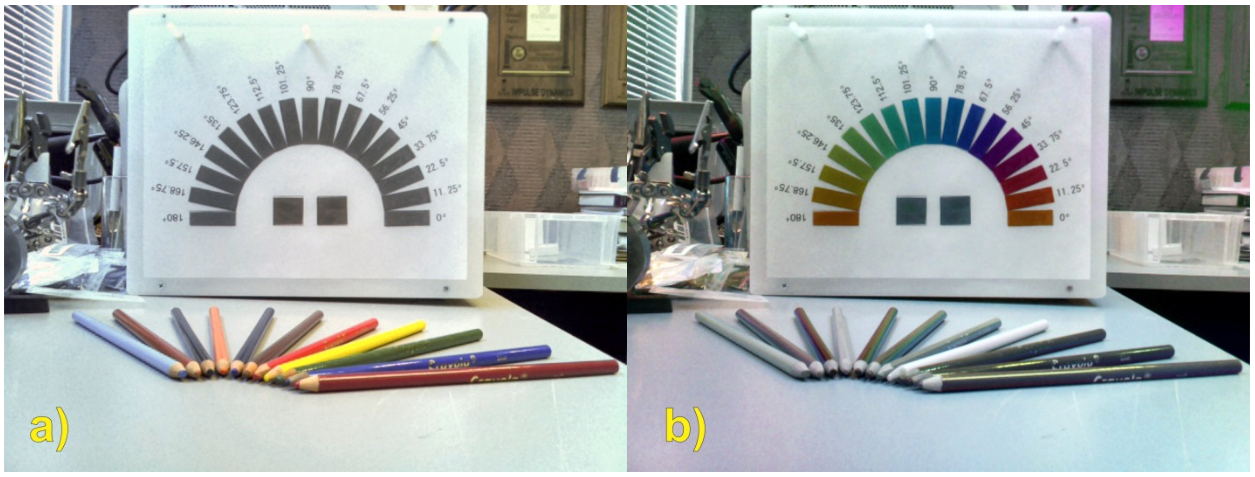


Figure 12 – Sample image obtained with DOLPi-Mech and the Python code of Table 2. (a) Unfiltered image (RasPi camera looking through open slot in filter wheel). (b) Picture taken by DOLPi-Mech by assigning the 0^{o} grayscale image to the Red channel of the output image, the 90^{o} grayscale to Blue, and the 45^{o} grayscale to Green. The colors in (b) have nothing to do with the actual color of the object, but instead encode the light's Angle of Polarization.

DOLPi's Electro-Optic Polarization Modulator

Mechanically-rotating a polarization analyzer (or a filter wheel as in DOLPi-Mech) is slow, which is not amenable to polarimetric video imaging. For this reason, the heart of the DOLPi camera is an Electro-Optic (EO) polarization modulator. This assembly, which consists of two liquid-crystal panels and a linear polarizer film, selectively passes light at different polarization angles under software control.

The liquid crystal panel used for the DOLPi camera is harvested from a low-cost auto-darkening welding mask filter (Figure 13). These are similar in construction to an LCD display, where the whole filter is a single gigantic "pixel". In its intended use, the auto-darkening filter allows the user to see through the filter while setting up the pieces to be welded, but as soon as light sensors detect the welding arc, the filter turns dark to protect the welder's eyes.

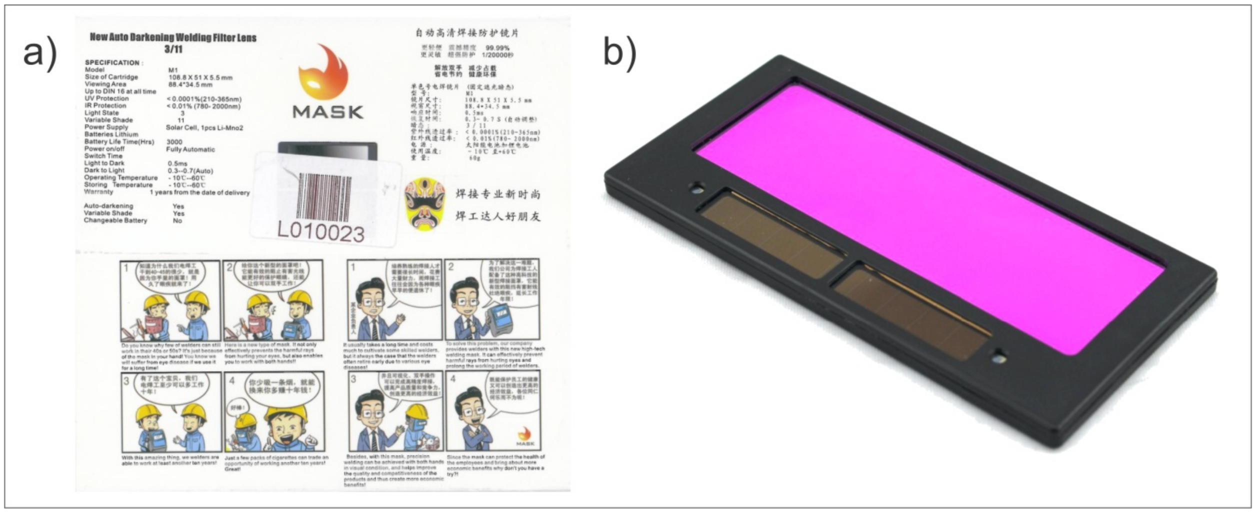


Figure 13 – The liquid crystal panel (LCP) for the DOLPi camera is harvested from a low-cost auto-darkening welder mask filter. I purchased these units for $9 each on eBay®. The box in which the filter came indicates that it is manufactured by "Mask" in China. Their website is www.auto-mask.com. b) The front side of the filter's enclosure (facing the pieces being welded) exposes the UV/IR filter that preceded the electrically-controlled light shutter, two photovoltaic panels ("solar cells"), and two light sensors used by the circuit to detect the presence of a welding arc.

The electrically-controlled light shutter in the auto-darkening filter is based on a liquid crystal panel (LCP) sandwiched between two polarizers. The liquid crystal panel is constructed from two glass windows separated a few microns apart. Each window is coated with transparent conductive Indium Tin Oxide (ITO) that serves as an electrode. Depending on the type of liquid crystal used, a thin dielectric layer may then applied over the ITO and gently rubbed to provide for liquid crystal molecular alignment. The cavity is then filled with birefringent nematic liquid crystal material[7]. Lastly, electrical contacts are attached and the panel is environmentally sealed.

As shown in Figure 14-a, with no bias voltage applied between the electrodes, the liquid crystal molecules are guided by the scratch lines rubbed to form the alignment layer, and thus align parallel to the windows. In this orientation the liquid crystal panel shifts the polarization of incoming light by (approximately) 90^o. However, as shown in Figure 14-b, when 5V are applied to the panel, the liquid crystal molecules align with the electric field and tip perpendicular to the windows. In this state, the polarization of incoming light remains largely unchanged (i.e. polarization shift is approximately 0^o).

[7] "Liquid crystals" are matter in a state that has properties between those of conventional liquid and those of solid crystal.

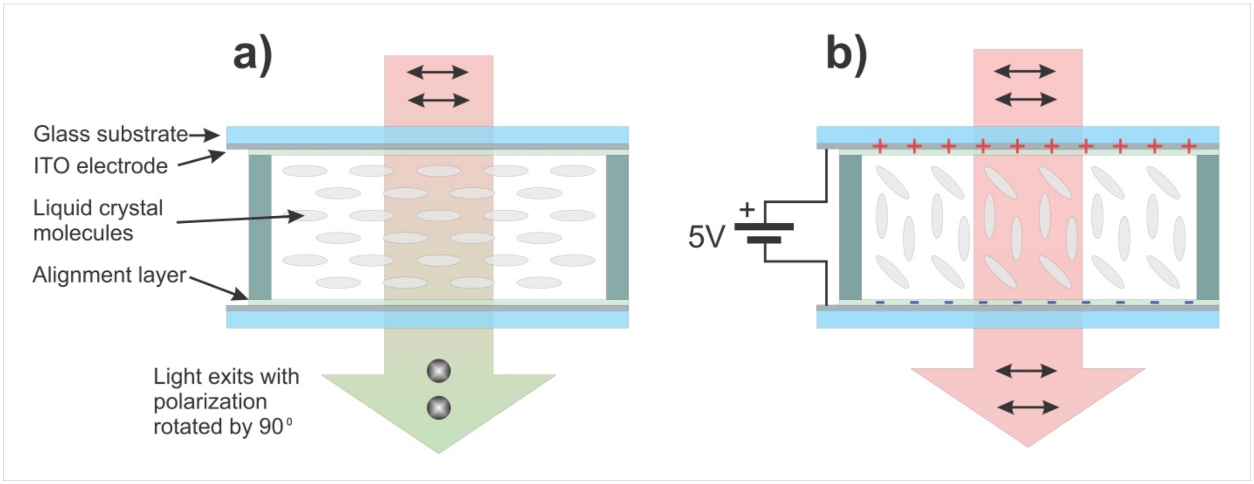


Figure 14 –Without a voltage applied to the LCP, the liquid crystal molecules have an ordered orientation, which together with the stretched shape of the molecules causes them to shift the polarization of light by 90^o (a). When an electric field is applied, the molecules align to the field and the polarization shift depends on the tilting of the liquid crystal molecules. Beyond some voltage, the LCP introduces almost no polarization tilt (b).

Figure 15 shows how the electrically-controlled light shutter in the auto-darkening filter works. Light from the scene is polarized by a first polarizer film. In the absence of the LCP, the polarized light would not be able to go through the second polarizer because it is set orthogonally with respect to the first polarizer. However, when the unbiased LCP is placed in the middle, it rotates the polarization of the light passing through the first polarizer by 90^o such that it can go through the second polarizer. However, to darken the filter, the mask's control circuit applies >5V to the LCP, which causes light going through it to maintain the polarization set by the first polarizer, thus preventing the light to go through the second polarizer and into the welder's eyes.

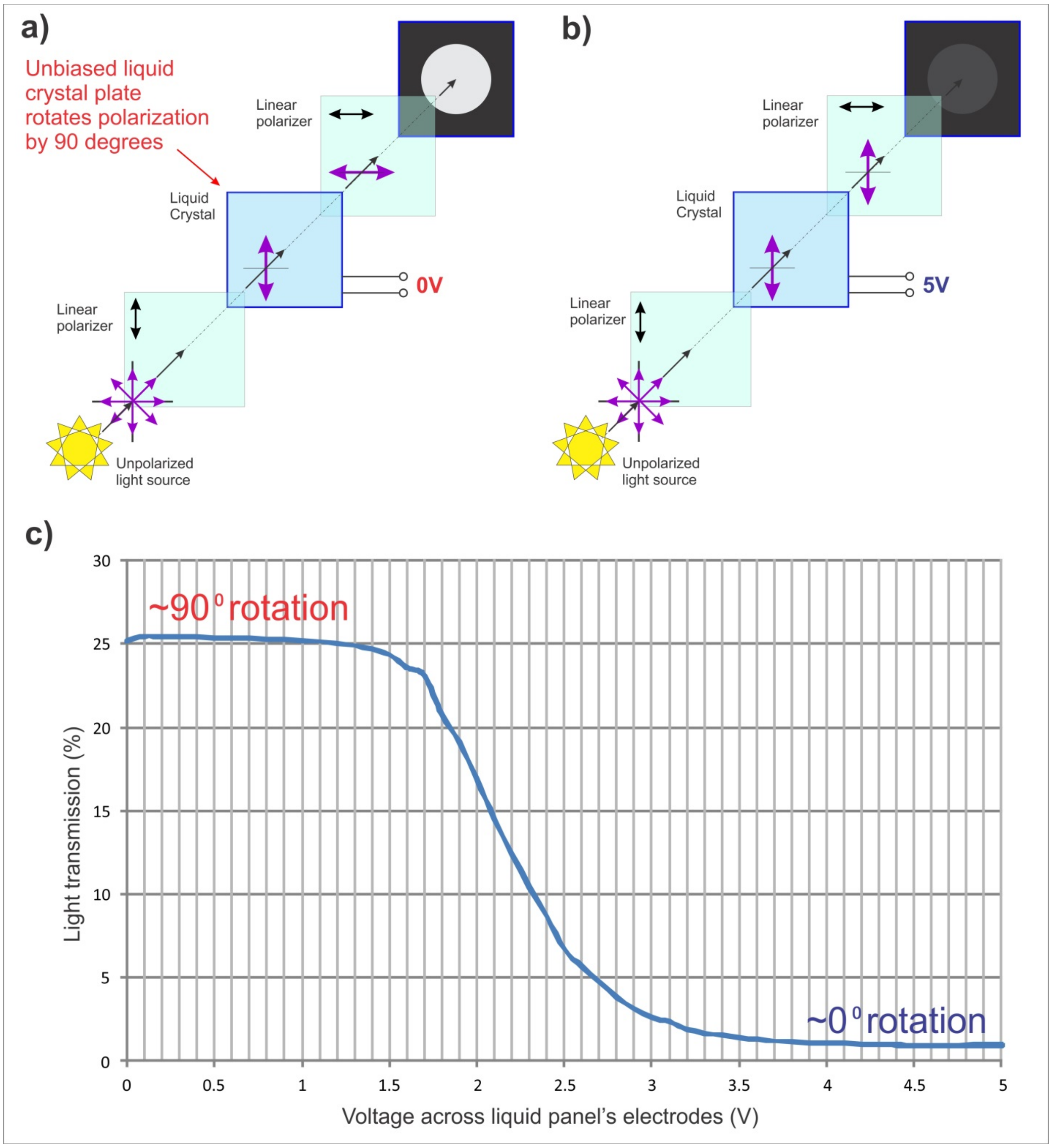


Figure 15 – The core component of the auto-darkening welding mask filter is a liquid crystal panel (LCP) sandwiched between two crossed polarizers. a) In the absence of an electric field, the LCP turns the polarization by 90°, so light can go through the second polarizer unimpeded. b) When 5V are applied across the LCP, polarization is conserved, so light cannot pass through the second polarizer. c) Transmission curve for the filtered, electrically-controlled light attenuator of the auto-darkening welding mask filter. Without the UV/IR filter (purple glass cover) the transmission at 0V is around 30% and drops to around 1% at 5V.

As shown in Figure 16, opening the plastic enclosure is easy. The two half-shells forming the enclosure are glued together and can be separated with a blade or screwdriver. The LCP's drive wires can then be unsoldered (or cut) from the circuit.

One side of the electro-optical assembly is purple, while the other is green. The purple glass is a filter that cuts out infrared and ultraviolet emissions from reaching the welder's eyes. This filter also cuts out some of the light reaching the polarizers and LCP. It can be removed with ease by cutting the bit of transparent tape on the sides and lifting it with a sharp razor. Without this filter, the unbiased optical attenuator passes around 30% of incoming light, and darkens to allow only 1% of the light through at a drive voltage of 5V. Go ahead – connect a variable-voltage power supply to the assembly and play with it. You should see that it lets light through up to around 1.5V when it starts to darken, reaching full attenuation at around 3.6V. The panel should tolerate voltages of around 10V with no damage.

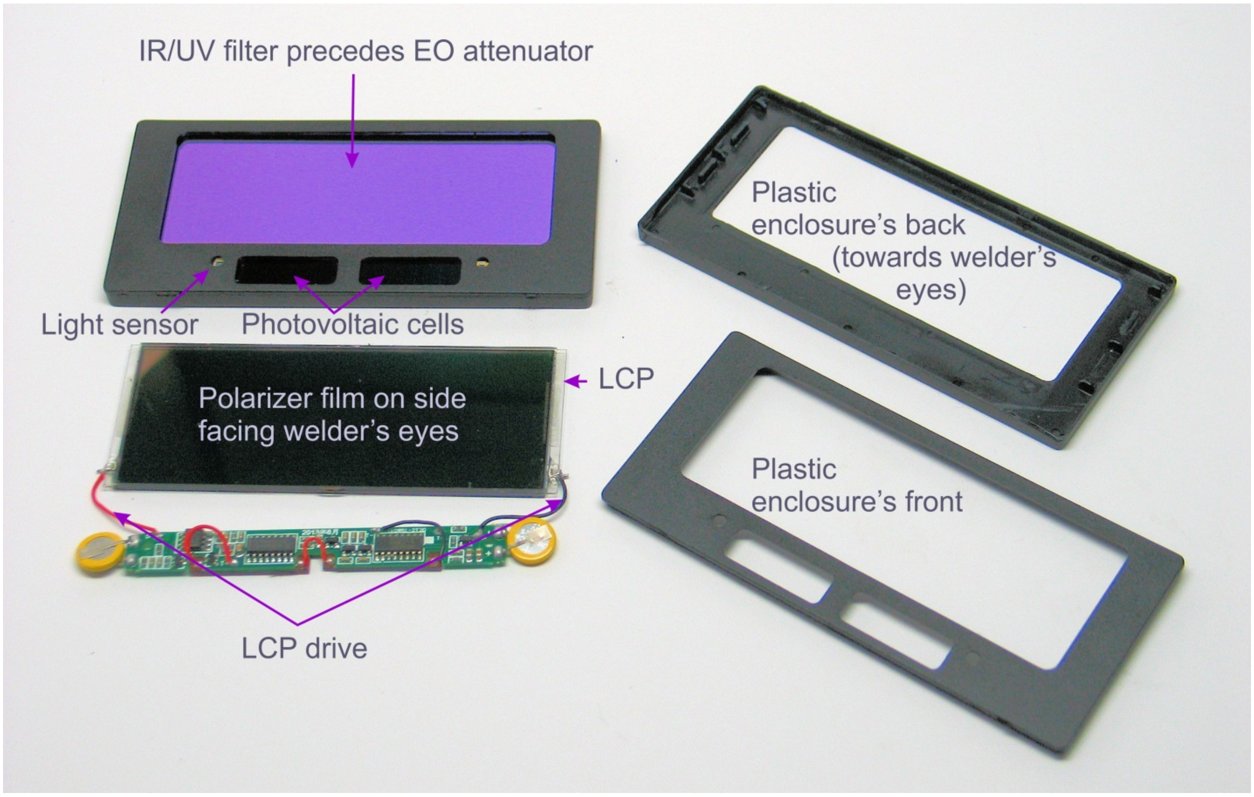

IR/UV filter precedes EO attenuator
Plastic enclosure's back (towards welder's eyes)
Light sensor
Photovoltaic cells
LCP
Polarizer film on side facing welder's eyes
Plastic enclosure's front
LCP drive

Figure 16 – Teardown of the auto-darkening welding mask filter. The electrically-controlled light shutter and its control circuit can be removed by prying apart the two half shells that form the plastic enclosure.

Polarization Imaging

One last step before being able to use the LCP for imaging is to remove the back-side polarizer film. It peels off easily after lifting a corner with a sharp hobby knife. The remaining polarizer and LCP now form a voltage-controlled polarization analyzer. Let's call this assembly a "VCPA" for short. It is important to note that the polarizing film used in auto-darkening welding masks is oriented at 45^o with respect to the edges of the LCP.

A very simple, yet interesting first experiment in polarization imaging can already be conducted at this stage with the VCPA. All you'll need besides the VCPA assembly is a 3Hz square-wave signal generator. An example circuit is shown in Figure 17.

Now, LCPs are commonly not operated with DC voltages for extended periods of time because DC polarization degrades the liquid crystal. In fact, critical-use drivers should have no net DC bias present in the drive signal in order to prevent ion migration within the LCP which impairs its performance and lifetime. Instead, LCPs should be driven with alternating current in the few kHz range. The LCP will respond to the RMS voltage across its electrodes. The LCP in the auto-darkening welding mask filter that I used is DC-driven, so this LCP may be DC-tolerant, but my recommendation is not to use the driver of Figure 17 for long periods of time.

Take the polarizer film that you pulled before from the assembly and set it some distance away from you. Now look at the scene through the VCPA (with the polarizer facing your eyes). While most of the scene should remain steady, the piece of polarizer will appear flashing when viewed through the VCPA. Try looking at items that include an LCD display (e.g. computer monitor, smartphone, calculator, etc.), as well as a piece of plastic reflecting sunlight through this device. Even with this simple gadget you should be able to appreciate the potential of adding polarization information to our regular color/intensity vision!

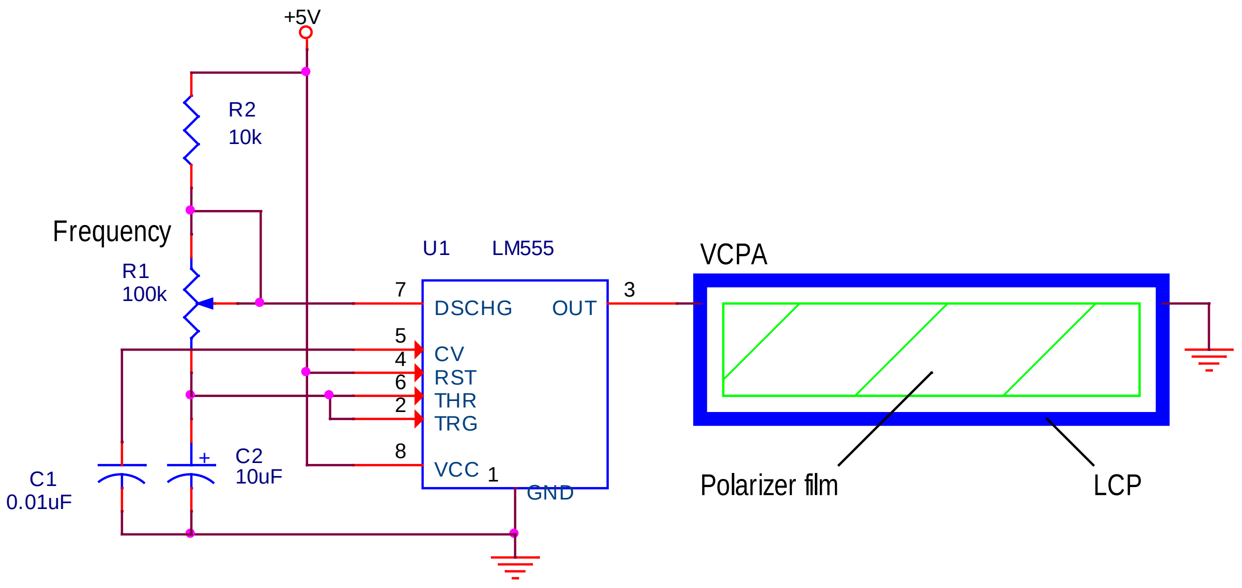


Figure 17 – A simple 555-based square-wave oscillator can be used to drive the VCPA at a frequency of around 3Hz. Materials that reflect polarized light appear to flash brightly when viewed through this device. Please see Appendix VI for a productized version of this circuit.

Boyd B. Bushman (of Area 51 UFO "deathbed confession" fame) described in a 1995 Patent (U.S. Patent 5,404,225) what essentially amounts to binoculars fitted with a VCPA to help detect man-made (or alien?[8]) objects, such as military vehicles and high-voltage power lines. The idea is that since these objects incorporate plenty of glass, plastic, paints, rubbers, and other materials that polarize light (Figure 19), they look different through a polarizer that is set at different angles [Driver and Fowler, 1967]. When viewed through the oscillator-driven VCPA, such materials will appear to flash.

Since most natural backgrounds do not show the polarizing contrast between angles of polarization, changing the angle of the polarization analyzer will not cause natural objects and background to vary. Only polarizing materials and surfaces will be highlighted, making targets stand out, giving away their location. Bushman reported that neither camouflage nor moderate foliage stops the systems from highlighting man-made targets because adequate flashing can still be observed.

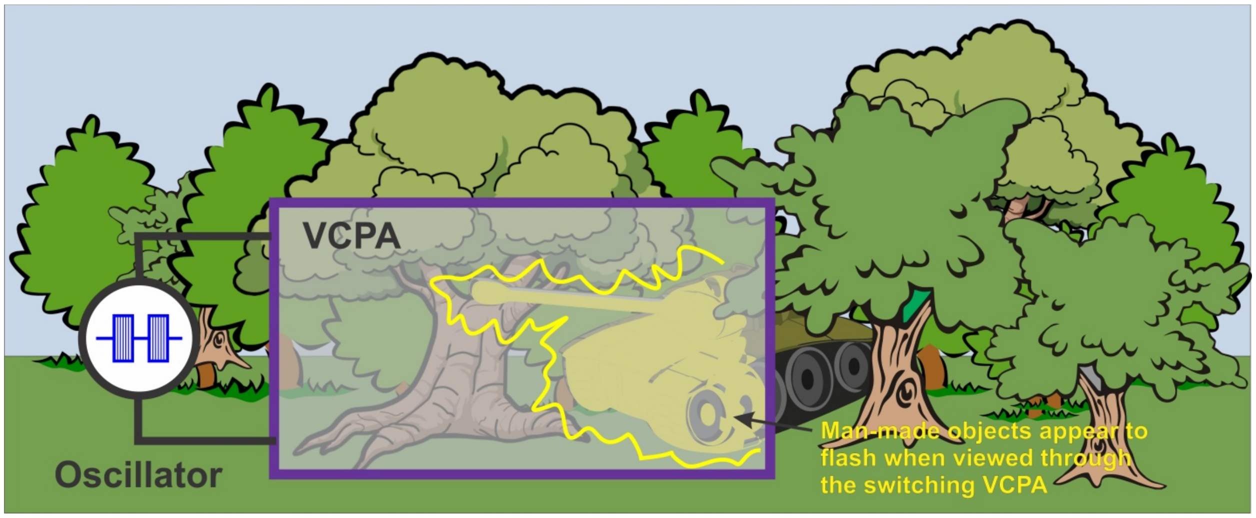

VCPA
Oscillator
Man-made objects appear to flash when viewed through the switching VCPA

Figure 18 – Man-made objects are easier to discriminate from a natural background if polarization information is used to supplement the image. A 1995 patent describes a pair of binoculars that incorporate a switching polarization analyzer to highlight military targets by making them flash against a steady background. Please see Appendix VI for a productized visor implementing this concept.

[8] I have to wonder if the true background behind this line of Boyd Bushman's research related more to his personal fascination with UFOs than with any true Lockheed military project.

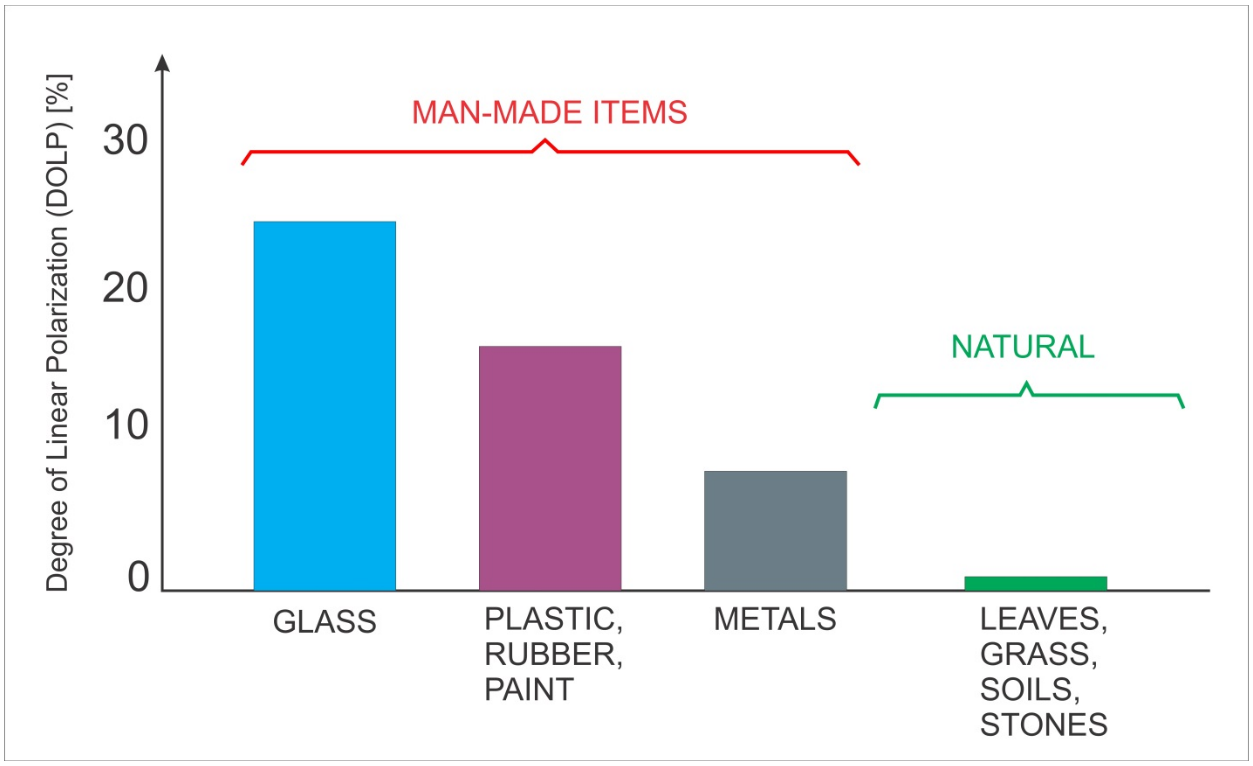


Figure 19 – Degree of Linear Polarization of the sunlight reflected by various types of man-made and natural materials. Adapted from Bushman BB, *Object Detector*, U.S. Patent 5,404,225, Apr. 4, 1995.

The same concept can be applied to polarization imaging with a camera, in which case the degree of polarization can be encoded by subtracting the image obtained with one polarization from a frame acquired with the polarization analyzer set at the orthogonal polarization. Of course, switching of the polarization analyzer in this case must be done synchronously with the camera's scan. This is the operating principle of the "Samba" polarimetric camera made by Bossa Nova Technologies of Culver City, CA., which provides "polarization contrast" images at video rate.

Polarization analysis with a single VCPA is known as "incomplete" because it cannot differentiate light polarized at 45^o with respect to the polarizer film in the VCPA as being polarized. Looking back at Figure 7-c we can see that light polarized at 45^o will look the same whether the analyzer is set to 0^o or 90^o. To deal with this condition, Bossa Nova Technologies added an optical component known as a "quarter-wave plate" and a second liquid crystal cell to rotate polarization by 45^o. Their "Salsa" camera is thus able to perform "complete" linear polarization analysis by switching both liquid-crystal plates through all four combinations to yield images analyzed at -45^o, 0^o, 45^o, and 90^o [Vedel et al., 2011; Lefaudeux, 2011]. This system uses custom-made ferroelectric liquid crystals mounted directly in front of the camera's CCD to achieve excellent linearity at 28 polarization frames per second at 320x240 resolution, or 8.75 frames per second at 782x582 pixel resolution.

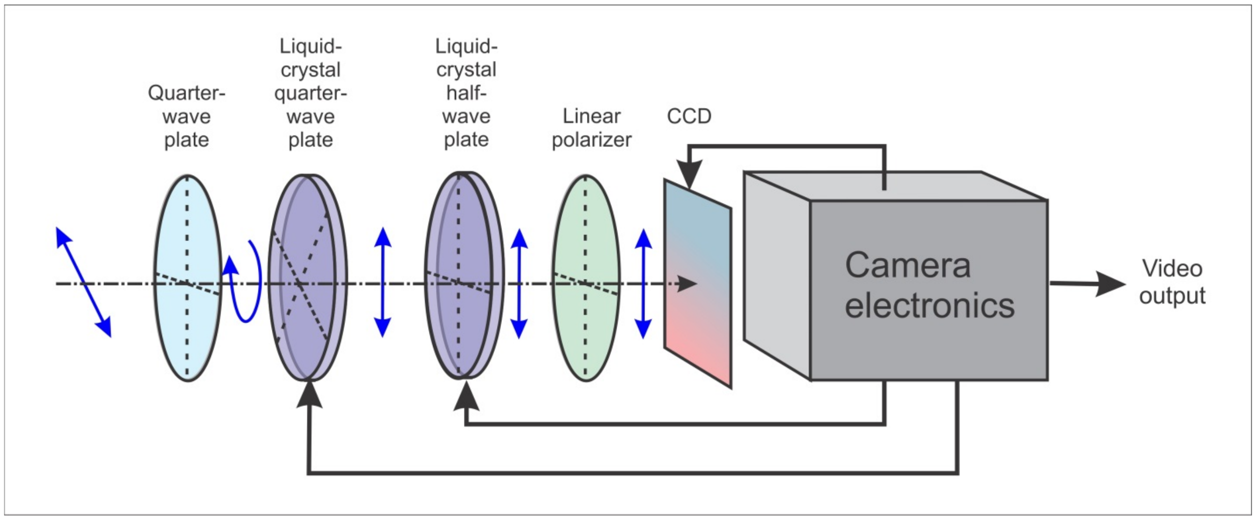


Figure 20 – Bossa Nova's Salsa camera achieves complete polarization imaging by switching a liquid-crystal quarter-wave plate preceded by a fixed quarter-wave plate, and a liquid-crystal half-wave plate followed by a linear polarizer through all four liquid-crystal setup combinations to yield video images containing information from individual frames analyzed at -45^o, 0^o, 45^o, and 90^o.

The DOLPi Camera

Bossa Nova's method is straightforward if laboratory optical-grade components are used. These are very expensive and out of reach for most private enthusiasts. However, I found through experimentation that a welding mask LCP and a polarizer sheet can also give very satisfactory results for qualitative analysis of linearly-polarized light.

The welding mask's LCP can be made to rotate polarization[9] to values between its two extremes at 0^o and 90^o. Although it is possible to find a voltage that will set the LCP to rotate light's polarization by around 45^o, changes in the liquid crystal take place with time, so the polarization state will shift with complex, interacting time constants that are dependent on temperature, age, and other factors. For this reason, the drive amplitude to achieve 45^o rotation of polarization needs to be periodically recalibrated.

This is a good place to state that not all liquid-crystal shutters are suitable for conversion into LCPAs that can be driven to analyze at 45^o. I tested modified light shutters from 3D active glasses made for the Sony PlayStation, as well as some low-cost universal 3D glasses for DLP projector units ("model G15-DLP") without success. The issue with these is that their liquid crystal panels have only two stable states (0^o and 90^o) with an extremely sharp transition zone that does not lend itself to intermediate

[9] This statement is not strictly true. The retardation process performed by a LCP is not the same as rotation of linear polarization. In reality, the LCP driven half-way acts as a quarter-wave plate, and hence the strict interpretation of the analysis at this level is for circular polarization rather than linear polarization at 45^o.
I didn't want to go into a thorough explanation of polarization optics to keep the project accessible, but based on my experiments, I'm convinced that DOLPi's "45^o image" indeed contains a dominant 45^o component when observing linearly polarized light. However, as we'll see later, DOLPi incorrectly understands circularly-polarized light as a 45^o signal.

polarization states. At the same time, I would like to mention that the only welding mask filter that I have tried is the one shown in Figure 13, and I don't know if LCPs from other auto-darkening welding mask filters would work equally well.

Figure 23 shows a block diagram for the DOLPi camera, which is my solution to acquiring the three pictures (analyzed at 0^o, 45^o, and 90^o) to perform complete linear polarization imaging of a scene. The imaging element is the official Raspberry Pi camera connected directly to a Raspberry Pi 2's dedicated camera connector. The camera views the scene through the VCPA hacked from an auto-darkening welding mask filter as discussed above. In my first prototype (Figure 27), I chose to tilt the VCPA by 45^o with respect to the camera's horizon because the polarizer film used in the mask is oriented at this angle, so tilting reorients it vertically with respect to the camera. However, as you will soon realize, identical results are possible when mounting the LCPA horizontally, vertically, or at 45^o by defining the correct frame of reference through software.

As discussed before, liquid crystal panels should not be operated with DC for prolonged periods of time, but if you want to first see what the camera can do before building a more complex AC driver with self-calibration, you can start by assembling the really simple circuit of Figure 21. The GPIO pins refer to the Raspberry Pi 2's GPIO interface.

When GPIO pin 21 and GPIO pin 22 are low, both LCP electrodes are at the same potential, and thus the LCP causes a polarization shift of 90^o. When GPIO 22 is high, the LCP is biased at +5V (minus a diode drop), causing no rotation in polarization (0^o). When GPIO 22 is low and GPIO 21 is set high, the LCP is biased at a voltage determined by potentiometer R1, which is manually set to produce maximum contrast in the DOLPi image between polarizers oriented at 45^o and -45^o.

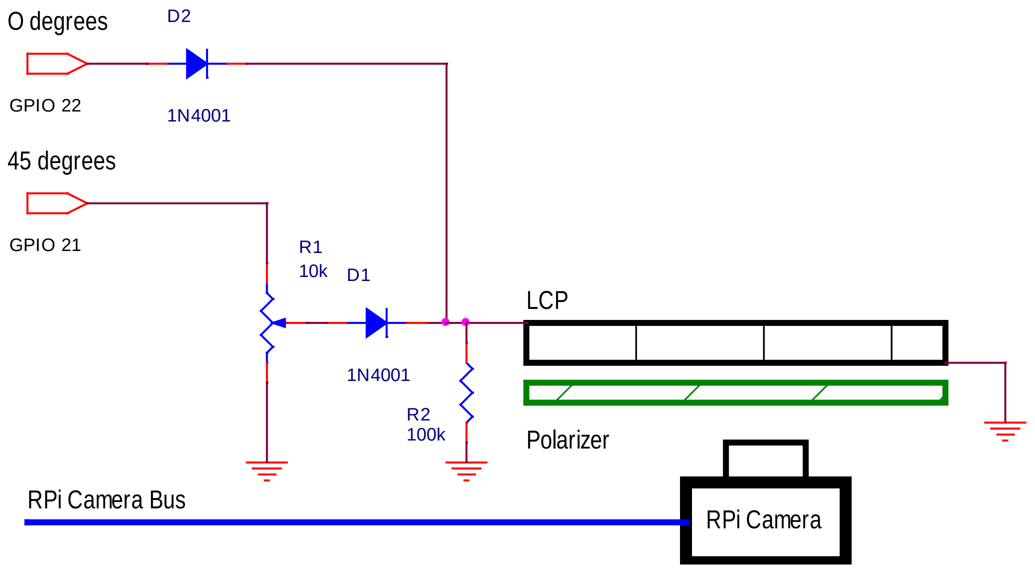
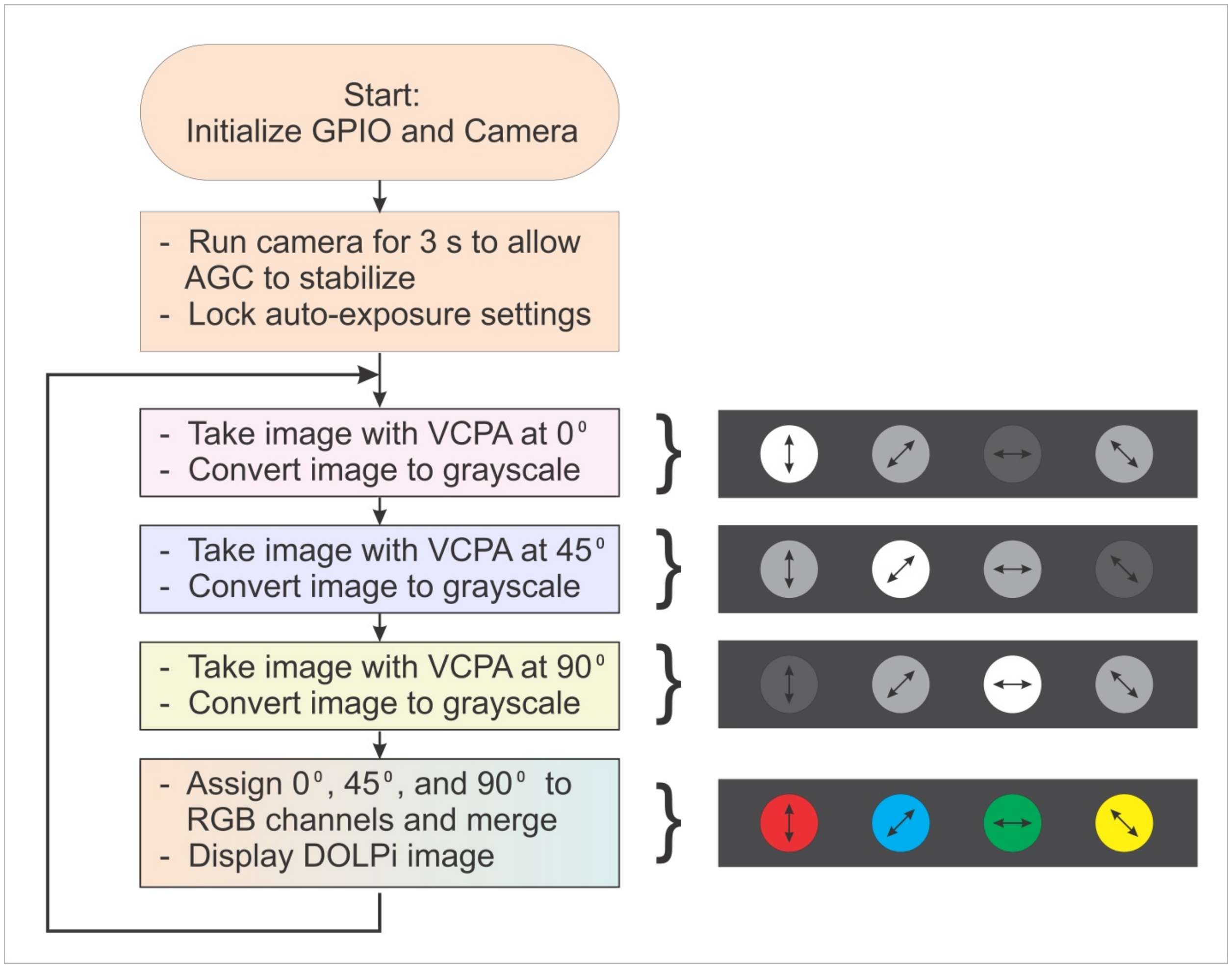


Figure 21 – The Raspberry Pi can drive the VCPA through this simple circuit to analyze at 0^o, 45^o, and 90^o. The voltage used to analyze at 45^o is selected manually via potentiometer R1. This circuit imposes DC polarization across the LCP, so it shouldn't be used for extended periods of time. However, it will quickly get you playing with polarimetric imaging until you get around to building the AC driver (Figure 24).

The Python listing for this simple polarization camera is shown in Table 3. Figure 22 shows a simplified flowchart for the software. The program first imports necessary packages and then allows the camera to sample the scene before locking its exposure settings. The imaging loop consists of taking three grayscale images with the VCPA set at 0°, 45°, and 90°, and then combining these pictures into a single color image encoding the scene's intensity and polarization.



Start:
Initialize GPIO and Camera
- Run camera for 3 s to allow AGC to stabilize
- Lock auto-exposure settings
- Take image with VCPA at 0°
- Convert image to grayscale
- Take image with VCPA at 45°
- Convert image to grayscale
- Take image with VCPA at 90°
- Convert image to grayscale
- Assign 0°, 45°, and 90° to RGB channels and merge
- Display DOLPi image

Figure 22 - This flowchart explains the simple software used to image with the camera of Figure 21. After loading libraries and initializing the Raspberry Pi's ports, the Raspberry camera is allowed to sample the scene for a few seconds to set its internal auto-exposure and then locking the exposure settings to keep consistency among images. The imaging loop consists of taking three grayscale images with the VCPA set at 0°, 45°, and 90°, and then combining these pictures into a single color image that encodes polarization.

Table 3 – DOLPiManual.py Python code

```
#  DOLPiManual.py
#
#  This Python program demonstrates the DOLPi polarimetric camera
#  under manual control (potentiometer setting for analyzer at 45 degrees)
#
```

```
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                                              (MIT, opensource.org/licenses/MIT)
#
#   This version uses a voltage-controlled polarization analyzer (VCPA) that
#   consists of a liquid crystal panel(LCP) and a polarizer film.  The polarizer
#   film is placed between the LCP and camera.
#   The VCPA is driven by GPIO pins 22 and 23 (via diodes and a pot) to 3 states:
#   1. 5V for no rotation of polarization
#   2. ~45 degrees of rotation as set by potentiometer
#   3. 0V for 90 degrees of rotation
#
#import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import RPi.GPIO as GPIO
import numpy as np


#IO PINS
#-------
dc0degpin=22   #0 degree VCPA DC bias manual mode
dc45degpin=21  #45 degree VCPA DC bias manual mode
GPIO.setwarnings(False)  #Don't issue warning messages if channels are defined
GPIO.setmode(GPIO.BCM)
GPIO.setup(dc0degpin,GPIO.OUT)
GPIO.setup(dc45degpin,GPIO.OUT)
GPIO.output(dc0degpin,False)
GPIO.output(dc45degpin,False)

#Raspberry Pi Camera Initialization
#----------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
#camera.resolution = (1280,720)
camera.framerate=80
rawCapture = PiRGBArray(camera)
camera.led=False

#Auto-Exposure Lock
#-----------------
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
```

```
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain

#Initialize flags
loop=True  #Initial state of loop flag
first=True #Flag to skip display during first loop
video=True #Use video port?  Video is faster, but image quality is significantly
        #lower than using still-image capture

while loop:
   #grab an image from the camera at 0 degrees
   GPIO.output(dc0degpin,True)
   GPIO.output(dc45degpin,False)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image0 = rawCapture.array
   # Select one of the two methods of color to grayscale conversion:
   # Blue channel gives better polarization information because wavelengt
   # range is limited
   # True grayscale conversion gives better gray balance for non-polarized light
   R=image0[:,:,1]  #Use blue channel
   #R=cv2.cvtColor(image0,cv2.COLOR_BGR2GRAY) #True grayscale conversion

   #grab an image from the camera at 45 degrees
   GPIO.output(dc0degpin,False)
   GPIO.output(dc45degpin,True)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image45 = rawCapture.array  #Capture image at 45 degrees rotation
   # Select one of the two methods of color to grayscale conversion:
   # Blue channel gives better polarization information because wavelengt
   # range is limited
   # True grayscale conversion gives better gray balance for non-polarized light
   G=image45[:,:,1]
   #G=cv2.cvtColor(image45,cv2.COLOR_BGR2GRAY)

   #grab an image from the camera at 90 degrees
   GPIO.output(dc0degpin,False)
   GPIO.output(dc45degpin,False)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   # Select one of the two methods of color to grayscale conversion:
   # Blue channel gives better polarization information because wavelengt
   # range is limited
```

```
    # True grayscale conversion gives better gray balance for non-polarized light
    image90 = rawCapture.array  #Capture image at 90 degree rotation
    B=image90[:,:,1]
    #B=cv2.cvtColor(image90,cv2.COLOR_BGR2GRAY)
    imageDOLPi=cv2.merge([B,G,R])
    cv2.imshow("Image_DOLPi",cv2.resize(imageDOLPi,(320,240),interpolation=cv2.INTER_AREA))
#Display DOLP image
    GPIO.output(dc0degpin,False)
    GPIO.output(dc45degpin,False)
    k = cv2.waitKey(1)  #Check keyboard for input
    if k == ord('x'):   # wait for x key to exit
        loop=False

#   Prepare to leave
cv2.imwrite("image0.jpg",image0)
cv2.imwrite("image90.jpg",image90)
cv2.imwrite("image45.jpg",image45)
cv2.imwrite("RGBpol.jpg",cv2.merge([B,G,R]))
cv2.imwrite("image0g.jpg",R)
cv2.imwrite("image90g.jpg",G)
cv2.imwrite("image45g.jpg",B)
cv2.destroyAllWindows()
quit
```

Now that you had a peek preview of DOLPi's capabilities, go ahead and build DOLPi with its proper AC driver and auto-calibration. A block diagram is shown in Figure 23. The VCPA is driven with 2 kHz square-wave AC. The signal is generated by a 555-based square wave oscillator and fed to an AC-coupled amplifier through a voltage-controlled resistor implemented from a simple P-type JFET. The amplifier boosts the drive signal to a maximum peak-to-peak amplitude of 10V. The Raspberry Pi controls the amplitude of the drive signal through a MCP4725 digital-to-analog converter (DAC or D/A).

I built a self-calibrating mechanism for the camera by adding a blue LED, a 45^o polarizer, and a light sensor (a light-dependent resistor – or LDR) on a separate optical path across the VCPA. Before each use, the Raspberry Pi turns on the LED and measures the 45^o-polarized light transmitted across the VCPA as the drive amplitude is ramped up. An ADS1015 12-bit, 4-channel analog-to-digital converter (ADC or A/D) is used by the Raspberry Pi to generate a transmission plot from which the D/A setting for maximum transmission can be found. The maximum value corresponds to the drive setting to be used to acquire images at an analysis angle of 45^o. By the way, you can try connecting the DAC's output directly to the LCP. The auto-calibration system should work if you want to DC-drive the LCP from the DAC and thus avoid constructing the AC driving circuit, but please keep in mind the possibility that this will deteriorate the LCP over time.

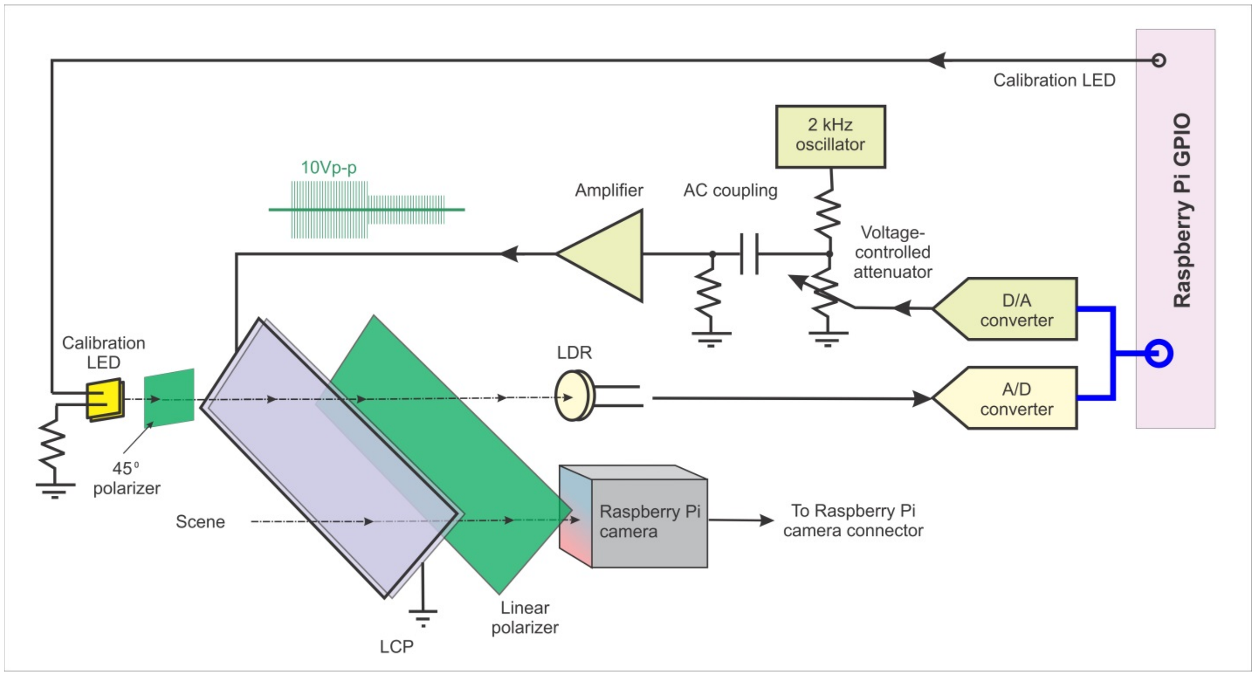


Figure 23 –Block diagram of the DOLPi camera. The VCPA is AC-driven at 2kHz. The drive amplitude is controlled from software via a D/A converter. The drive amplitude for 45° polarization shift is determined by an auto-calibration circuit consisting of an A/D converter that measures the intensity of 45° polarized light that goes through the VCPA as a function of D/A output.

Table 4 – Bill of Materials for DOLPi (some may be replaced or are optional)

Component reference	Component/Material	Source
RasPi	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	Adafruit ID:2358
SD Card	SanDisk SAEMSD64GBU3 64GB Extreme UHS-I microSDXC Memory Card (U3/Class 10)	B&H Photo
Keyboard/Mouse	Miniature Wireless USB Keyboard with Touchpad	Adafruit ID:922
RasPi Camera	Raspberry Pi Camera Board	Adafruit ID:1367
PiTFT	PiTFT Plus 480x320 3.5" TFT + Touchscreen for Raspberry Pi (Pi 2 and Model A+ / B+)	Adafruit ID:2441
VCPA	VCPA hacked from auto-darkening welding mask filter per text	Online (Amazon, eBay, etc). Filter manufactured by "Mask" www.auto-mask.com (see Figure 13)
ADC	ADS1015 Breakout Board - 12-Bit ADC - 4 Channel with Programmable Gain Amplifier	Adafruit ID:1083
DAC	MCP4725 Breakout Board - 12-Bit DAC w/I2C Interface	Adafruit ID:935
Battery Pack	USB Battery Pack for Raspberry Pi -	Adafruit ID:1565

	4400mAh - 5V @ 1A	
Power Switch	Breadboard-friendly SPDT Slide Switch	Adafruit ID:805
GPIO Stacking Header	Stacking Header for Pi A+/B+/Pi 2 - 2x20 Extra Tall Header	Adafruit ID:1979
D1	Diffused blue LED	Adafruit ID:780
R7	CdS photoresistor	Adafruit ID: 161
U1	TLC555CP LinCMOS 555 timer IC	Digikey 296-1857-5-ND
Q1	J112 small-signal P-type JFET	Mouser 512-J112
U2	LTC1152 rail-to-rail op-amp IC	Digikey LTC1152CN8#PBF-ND
U3	ICL7660 negative voltage converter IC	Digikey ICL7660CPAZ-ND
C3, C5	0.01µF capacitor	Radio Shack 272-131
C1	0.1µF capacitor	Radio Shack 272-135
C2	0.47µF capacitor	Mouser 81-RDER71H474K1M1H3A
C4, C6	10µF tantalum capacitor	Radio Shack 272-1436
R3	270Ω ½W resistor	Radio Shack 271-1112
R8	1kΩ ¼W resistor	Radio Shack 271-1321
R1	5.6kΩ ¼W resistor	Radio Shack 271-1125
R2, R4, R6, R9, R10	10kΩ ¼W resistor	Radio Shack 271-1335
R5	1MΩ ¼W resistor	Radio Shack 271-1134
SW1	Shutter release: Tactile Switch Button, 12mm square, 6mm tall	Adafruit ID: 1119
Sockets for U1, U2, U3	8-pin IC socket	Radio Shack 276-1995
DOLPi hat PCB	4.5x6.625" 2200-hole grid-style PC board	Radio Shack 276-147
Misc. hardware	Nylon standoffs, bolts/nuts, Delrin block for tripod mount, black plastic board, industrial-grade hook&loop tape, etc.	
Misc. prototyping	USB cable, wire, connector pins, bus wire, etc.	
Assorted polarizer films for testing	Polarizer educational kit	Edmund Optics 38490
Optional Manual Drive		
Figure 21 R1	Breadboard trim potentiometer 10kΩ	Adafruit ID:356
Figure 21 D1, D2	1N4001	Radio Shack 276-1101
Figure 21 R2	10kΩ ¼W resistor	Radio Shack 271-1335

A detailed schematic of the DOLPi camera is shown in Figure 24. The ADS1015 ADC and MCP4725 DAC are very small surface-mount parts, so I recommend purchasing them already mounted in breakout boards (subcircuits shown within dashed boxes) from Adafruit.com as models 1083 ($9.95) and 935 ($4.95), respectively.

I chose to use a cadmium sulphide light-dependent resistor (CdS LDR) instead of a more sophisticated light sensor because I wanted DOLPi to be easily reproducible using parts available at stores that cater to hobbyists. The specific model is not critical – an Adafruit model 161 ($0.95), Sparkfun model SEN-09088 ($1.50), or Radio Shack model 276-1657 ($3.99 for a pack of 5) will work equally well.

The LCP driver consists of a 555-type square wave generator running at 2kHz feeding an op-amp amplifier through a voltage-controlled attenuator and an AC-coupling capacitor. The J112 P-type JFET acts as a voltage-controlled resistor, which drops the 2kHz square wave's amplitude at point "A" depending on the voltage at its gate. The amplifier simply boosts the drive signal to a maximum peak-to-peak amplitude of 10V (±5V against ground). The negative power rail (-5V) for the op-amp is generated by an ICL7660 voltage converter from the Raspberry Pi's 5V power bus.

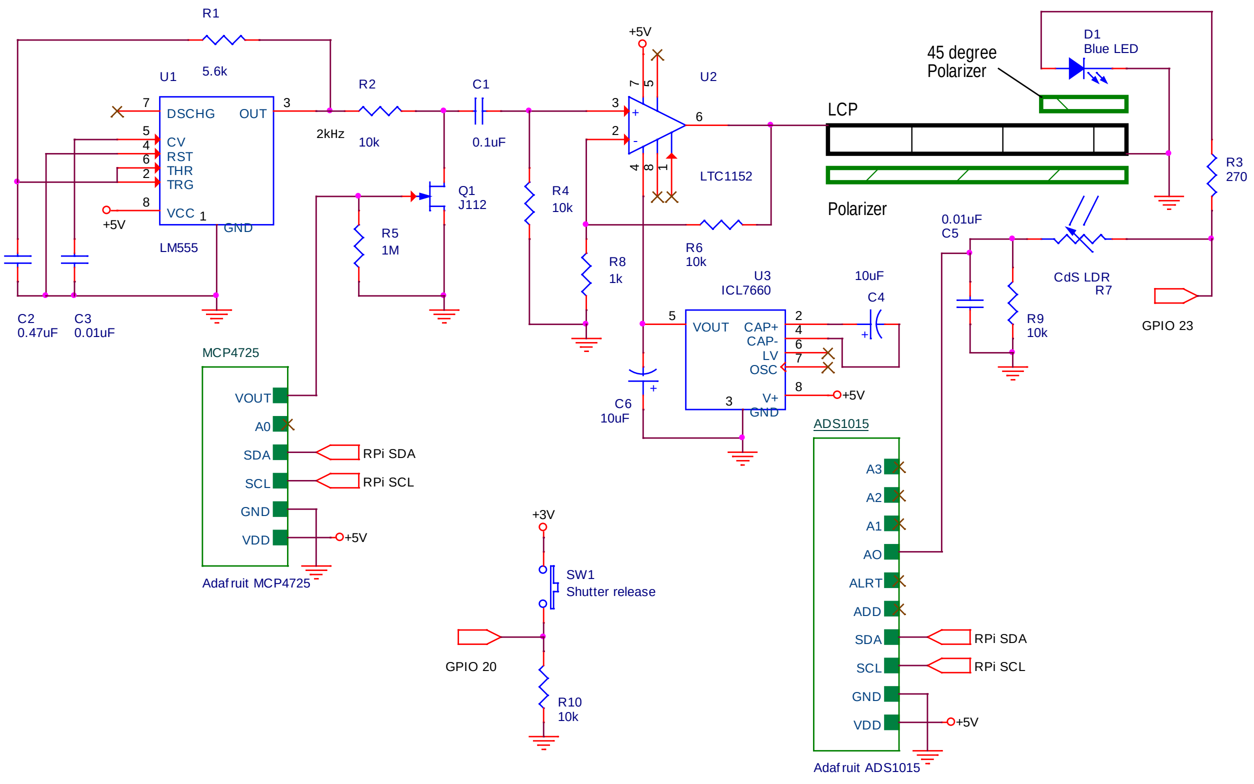


Figure 24 – Schematic diagram for the DOLPi VCPA driver and calibration circuit. The circuit is powered by the Raspberry Pi's +5V power rail. -5V are generated by U3 – a DC/DC negative voltage converter. This allows op-amp U2 to drive the LCP with a 2kHz (generated by U1) square wave that varies from +5 to -5V with respect to ground thus preventing ionic migration within the LCP (you may need to tweak the value of C2 and R1 to attain approximately 2kHz). R2 and Q1 act as a voltage-controlled attenuator such that the amplitude of the LCP's drive signal can be varied under the control of a MCP4725 DAC. The calibration circuit consists of a LED that illuminates a CdS light-dependent resistor (R7) through a piece of polarizer oriented 45^o with respect to the VCPA's polarizer. The voltage across R9 (which is proportional to the light illuminating R7) is read by an ADS1015 ADC. Table 5 shows the connections to the Raspberry Pi 2.

There is nothing critical about the construction of the driver circuit. I built my first prototype on a solderless breadboard and it worked great (Figure 27). Components could be substituted by similar parts. For example, although I used the TLC555 LinCMOS version of the 555, any other type of 555 timer IC should work in this circuit. Similarly, you could substitute the J112 for most any other small-signal P-type JFET, most any rail-to-rail op-amp should work fine instead of the LTC1152, and you could select any similar substitute for the LTC7660 (e.g. MAX1044).

Table 5 – GPIO header pin connections required by DOLPi

Raspberry Pi 2 GPIO Header Pin	Function	DOLPi Use
2, 4	+5V	+5V
6, 9, 14, 20, 25, 30, 34, 39	GND	GND
3	I^2C SDA	I^2C SDA to ADC and DAC
5	I^2C SCL	I^2C SCL to ADC and DAC
16	GPIO 23	Calibration LED and LDR power
15	GPIO 22	0^o DC Bias (optional)
40	GPIO 21	45^o DC Bias through potentiometer (optional)
38	GPIO 20	Shutter release pushbutton
23	SPI SCK	SPI Adafruit PiTFT 3.5”
19	SPI MOSI	SPI Adafruit PiTFT 3.5”
21	SPI MISO	SPI Adafruit PiTFT 3.5”
24	CE0	SPI Adafruit PiTFT 3.5”
26	CE1	SPI Adafruit PiTFT 3.5”
18	GPIO 24	Adafruit PiTFT 3.5”
22	GPIO 25	Adafruit PiTFT 3.5”
12	GPIO 18	Adafruit PiTFT 3.5” (PWM dim)

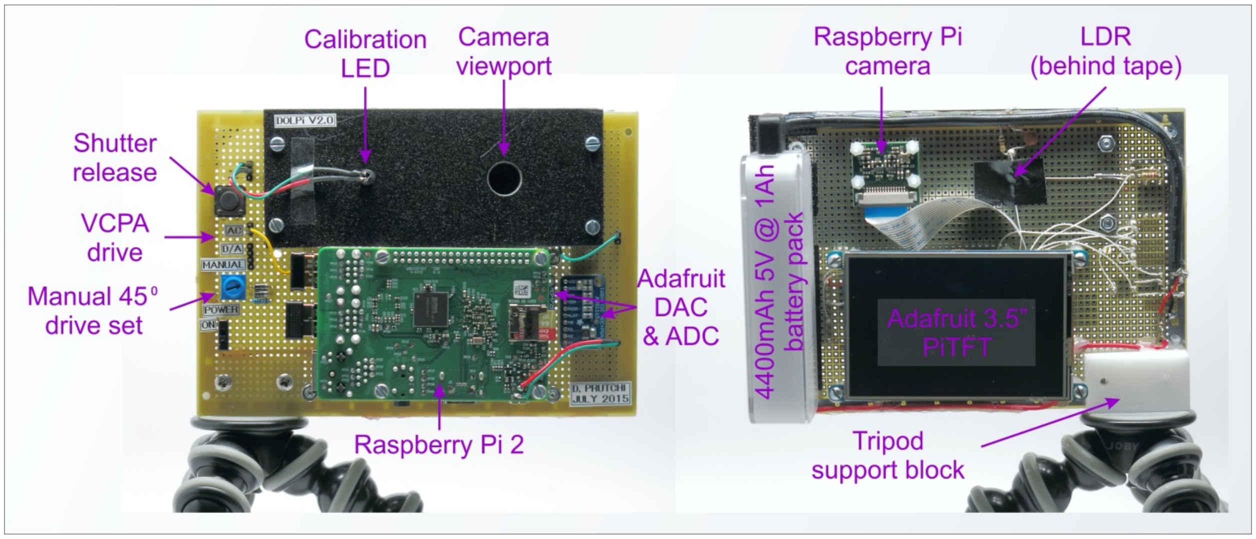


Figure 25 – I built my second DOLPi prototype on a piece of perfboard which not only serves as the substrate for the AC driver circuit, but also acts as a support platform for all other components.

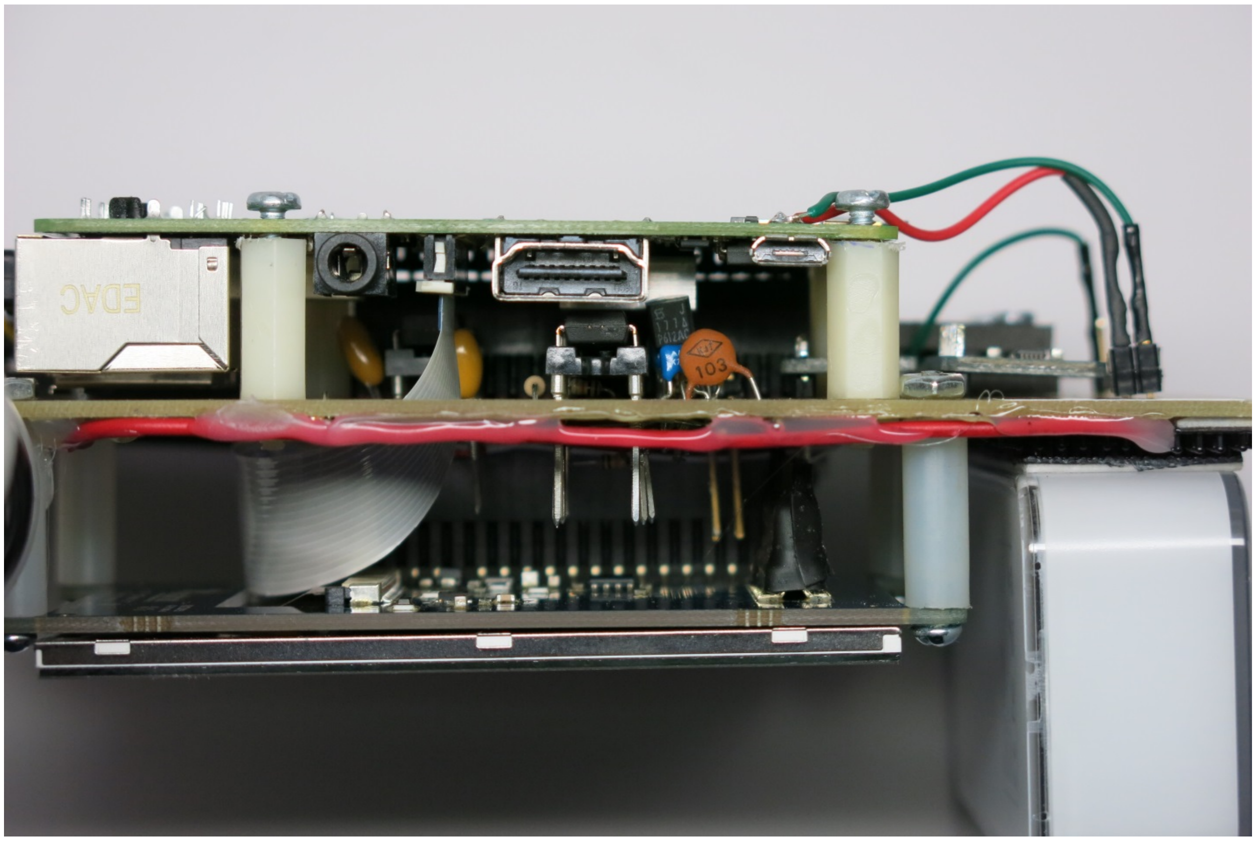


Figure 26 – The VCPA's AC driver circuit fits nicely under the Raspberry Pi 2. An Adafruit extra-long GPIO header extender is soldered to the perfboard and supplies power and control lines for both the DOLPi circuitry and PiTFT display directly from the Raspberry Pi.

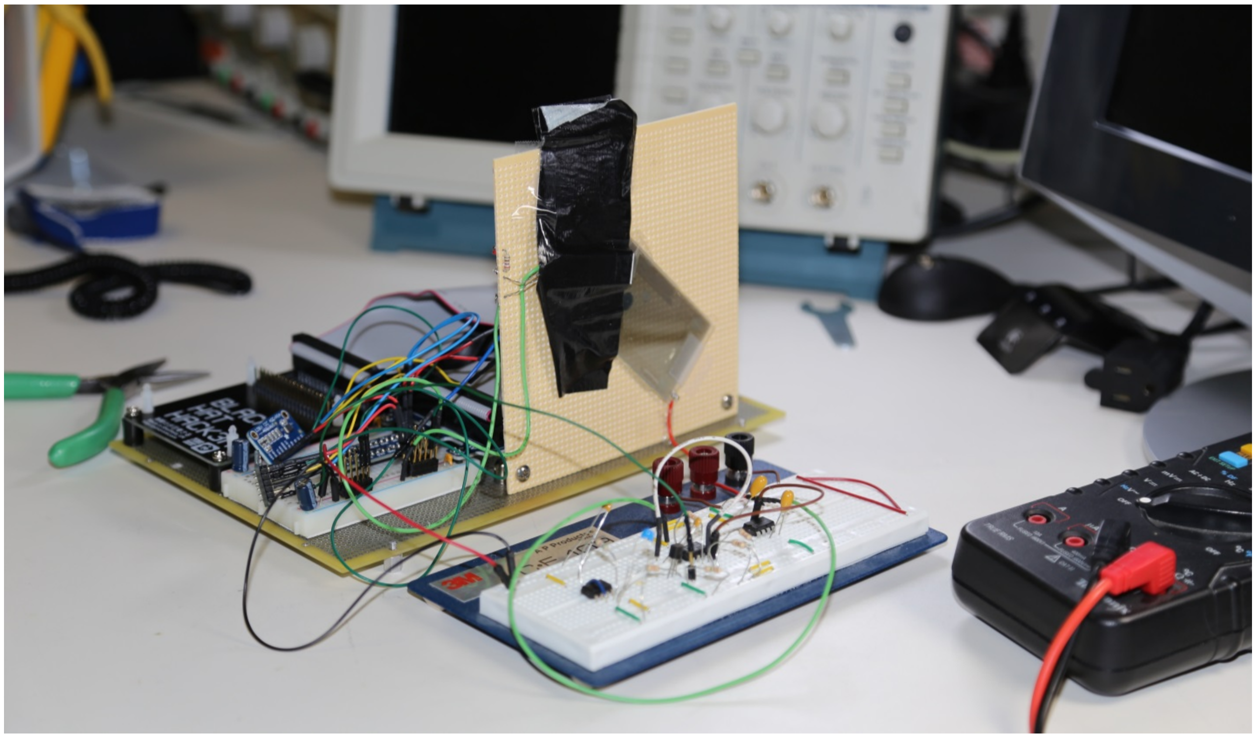


Figure 27 - There is nothing critical about the construction of the driver circuit, and components could be substituted by similar parts. I built my first prototype on a solderless breadboard and it worked great. The black duct tape holds the 45^o autocalibration polarizer and shields the LED and LDR from external light.

Although the circuit of Figure 24 works well, I'm not too happy with the intrinsic non-linearity of the FET. Because of this, an alternative, somewhat more complex, but I believe more elegant design is shown in Figure 28. I designed it so that its output will be perfectly linear with the DAC's output. The non-inverting amplifier built around U1 approximately doubles the output of the DAC. This voltage is then presented to an H-bridge implemented by the analog switches in U2. The H-bridge produces a continuous biphasic train at a frequency given by the oscillator built around U3A and U3B. U4 doubles the +5V coming from the Raspberry Pi to power U1 and U2. The LCP is connected between the legs of the H-bridge.

This circuit will drive most any LC panel (including lab-grade variable waveplates). However, the circuit can be simplified if you will use the LC panel harvested from the welding mask filter because it can be driven to 0^o shift with a smaller peak-peak square wave, so the op-amp and supply voltage doubler can be omitted. The final schematic for the driver is shown as part of Appendix V.

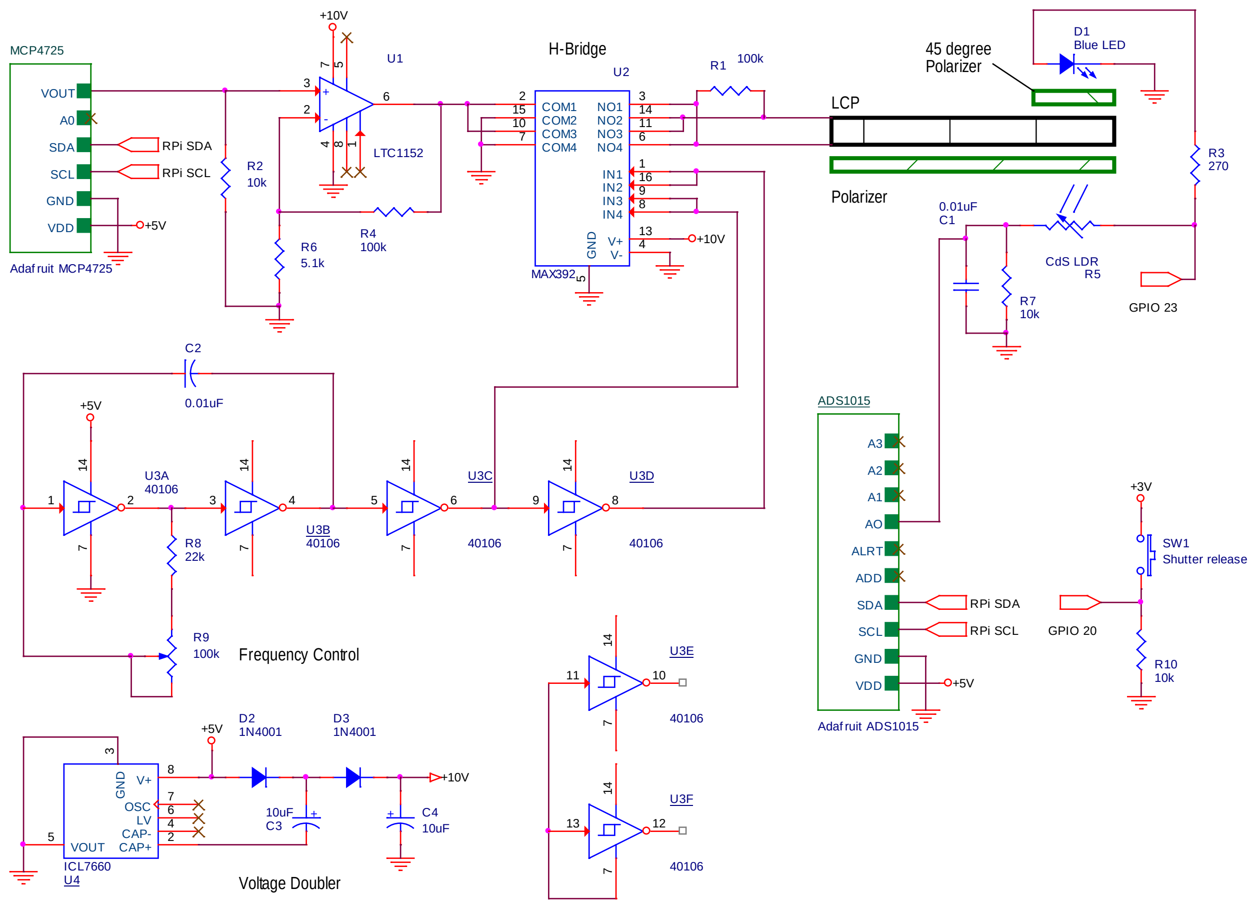


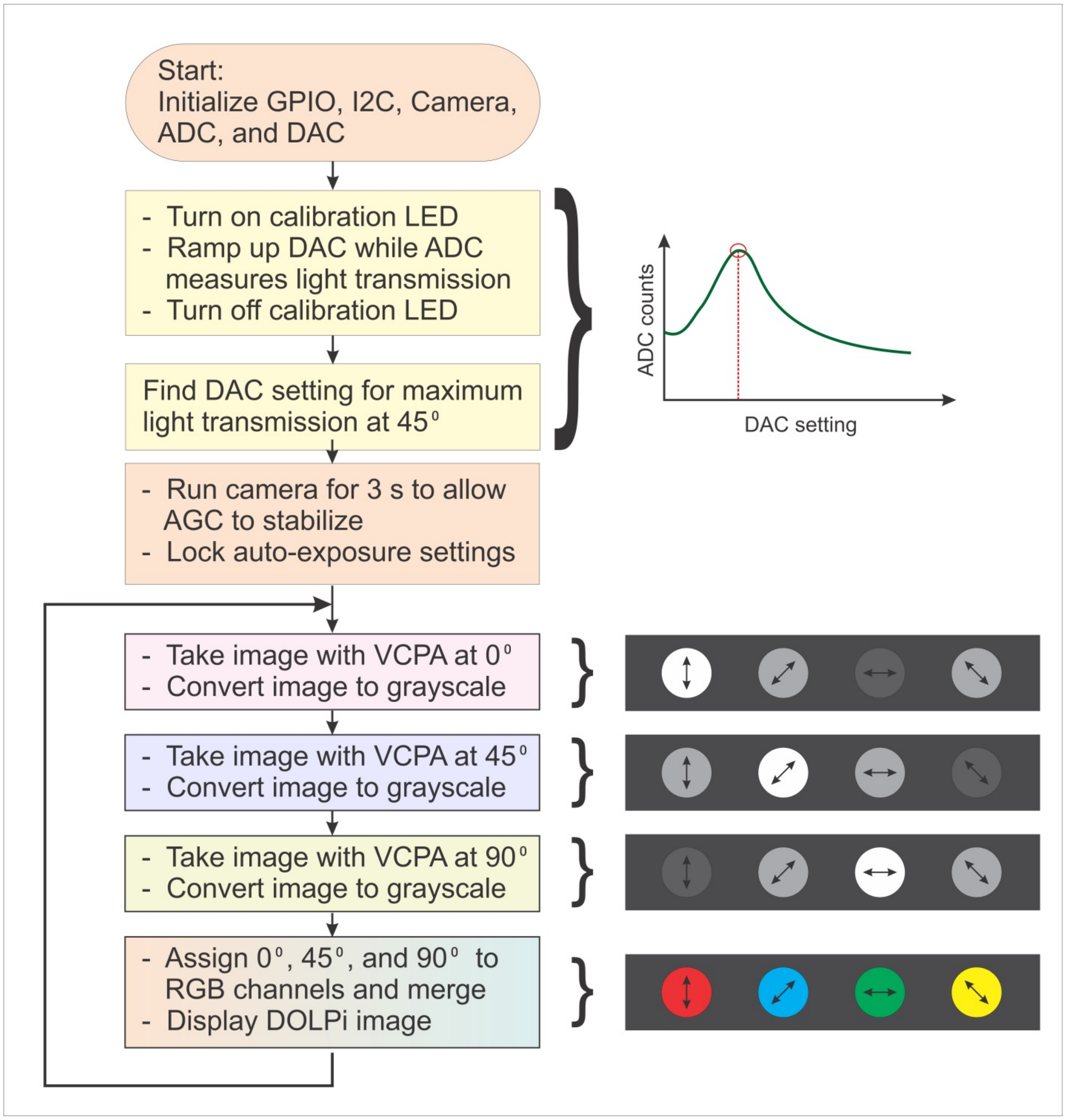
Figure 28 – This AC-driver circuit is more elegant than the one in Figure 24 because its output is perfectly linear with the DAC's output. The non-inverting amplifier built around U1 approximately doubles the output of the DAC. This voltage is then presented to an H-bridge implemented by the analog switches in U2. The H-bridge produces a continuous biphasic train at a frequency given by the oscillator built around U3A and U3B. U4 doubles the +5V coming from the Raspberry Pi to power U1 and U2.

The flowchart for the simplest automatic DOLPi software is shown in Figure 29. The Python code is shown in Table 6. After loading required libraries and initializing the Raspberry Pi's ports, the device runs the auto-calibration procedure by sweeping through LCP drive amplitudes between 0V and 10V$_{p-p}$ while measuring the intensity of 45^{o}-polarized light going through the VCPA. The DAC setting at maximum transmission is stored for use in taking frames at 45^{o}.

Like with the simple DC-biased camera software, the Raspberry camera is then allowed to sample the scene for a few seconds to set its internal auto-exposure and then locking the exposure settings to keep consistency among images. More on the methods available to capture consistent images with the Raspberry Pi's camera can be found at:

http://picamera.readthedocs.org/en/latest/recipes1.html#capturing-consistent-images.

Also like before, the imaging loop consists of taking three grayscale images with the VCPA set at 0º, 45º, and 90º, and then combining these pictures into a single color image encoding the scene's intensity and polarization.



Start:
Initialize GPIO, I2C, Camera, ADC, and DAC
- Turn on calibration LED
- Ramp up DAC while ADC measures light transmission
- Turn off calibration LED
Find DAC setting for maximum light transmission at 45°
ADC counts
DAC setting
- Run camera for 3 s to allow AGC to stabilize
- Lock auto-exposure settings
- Take image with VCPA at 0°
- Convert image to grayscale
- Take image with VCPA at 45°
- Convert image to grayscale
- Take image with VCPA at 90°
- Convert image to grayscale
- Assign 0°, 45°, and 90° to RGB channels and merge
- Display DOLPi image

Figure 29 – This flowchart explains the simple software used to auto-calibrate and image with the DOLPi camera. After loading libraries and initializing the Raspberry Pi's ports, the device runs the auto-calibration procedure by sweeping through LCP drive amplitudes between 0V and 10Vp-p while measuring the intensity of 45º-polarized light going through the VCPA. The DAC setting at maximum transmission is stored for use in taking frames at 45º. The Raspberry camera is then allowed to sample the scene for a few seconds to set its internal auto-exposure and then locking the exposure settings to keep consistency among images. The imaging loop consists of taking three grayscale images with the VCPA set at 0º, 45º, and 90º, and then combining these pictures into a single color image that encodes polarization.

Table 6 – DOLPiAuto.py Python code

```
#   DOLPiAuto.py
#
#   This Python program demonstrates the DOLPi polarimetric camera
#   under automatic control (DAC autocalibration for analyzer at 45 degrees)
#
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                                              (MIT, opensource.org/licenses/MIT)
#
#   This version uses a voltage-controlled polarization analyzer (VCPA) that
#   consists of a liquid crystal panel(LCP) and a polarizer film.  The polarizer
#   film is placed between the LCP and camera.
#   The VCPA is driven by a DAC-controlled AC driver to 3 states:
#   1. Highest DAC output of 5V for no rotation of polarization
#   2. ~45 degrees of rotation
#   3. Low DAC output for 90 degrees of rotation
#
#   An auto-calibration setup is used to determine the optimal voltages to
#   drive the VCPA.  The calibrator consists of a diffuse blue-light
#   LED illuminating the VCPA through a polarizer set at 45 degrees with
#   respect to the VCPA's polarizer.
#   Light transmission is detected by a LDR behind the VCPA and
#   measured by an ADS1015 Analog-to-Digital (ADC) converter IC
#
#import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import RPi.GPIO as GPIO
import numpy as np
from Adafruit_MCP4725 import MCP4725  #controls MCP4725 DAC
from Adafruit_ADS1x15 import ADS1x15  #controls ADS1015 ADC

#IO PINS
#-------
calLEDpin=23   #Calibration LED connected to pin 24
dc0degpin=22   #0 degree VCPA DC bias manual mode
dc45degpin=21  #45 degree VCPA DC bias manual mode
GPIO.setwarnings(False)  #Don't issue warning messages if channels are defined
GPIO.setmode(GPIO.BCM)
GPIO.setup(calLEDpin,GPIO.OUT)
GPIO.setup(dc0degpin,GPIO.OUT)
GPIO.setup(dc45degpin,GPIO.OUT)
GPIO.output(calLEDpin,False)  #Turn calibration LED off
GPIO.output(dc0degpin,False)
GPIO.output(dc45degpin,False)

#ADS1015 A/D DEFINITIONS
#-----------------------
# Select the A/D chip
#ADS1015 = 0x00  # 12-bit ADC
ADS1115 = 0x01              # 16-bit ADC

# Select the gain
#gain = 6144  # +/- 6.144V
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
#gain = 256   # +/- 0.256V

# Select the sample rate
#sps = 8    # 8 samples per second
# sps = 16   # 16 samples per second
# sps = 32   # 32 samples per second
#sps = 64   # 64 samples per second
# sps = 128  # 128 samples per second
# sps = 250  # 250 samples per second
# sps = 475  # 475 samples per second
sps = 860  # 860 samples per second

# Initialize the ADC using the default mode (use default I2C address)
# Set this to ADS1015 or ADS1115 depending on the ADC you are using!
adc = ADS1x15(ic=ADS1115)

#MCP4725 D/A DEFINITIONS
#-----------------------
# Initialize the DAC using the default address
dac = MCP4725(0x62)
# DAC output connected to frontLCD
dac.setVoltage(0)

def cal():
    #DRIVE VOLTAGE CALIBRATION
    #-------------------------
    # A/D converter is connected to a Light Dependent Resistor (LDR) placed
    # behind VCPA.  A LED illuminates the VCPA through a piece of polarizer film.
    # This function finds the drive voltages at which the VCPA should be driven
    # to analyze at 45 degrees
    #
    import matplotlib.pyplot as plt     #Needed only if plotting
    print "Please wait while calibrating..."
    #
    GPIO.output(calLEDpin,True)        #Turn calibration LED on
    #
    # Initialize LCP and flush A/D
    for i in range (1,5):
        dac.setVoltage(4095)          #Turn LCP full ON
        time.sleep(.1)                #Let it settle
        #dac.setVoltage(0)            #Turn LCP OFF
        flush=adc.readADCSingleEnded(0, gain, sps) #Flush the A/D
    #
    #Initialize the lists to hold the graph
    vol=[]                            #List to hold DAC voltage codes
    light=[]                          #List to hold light transmission
    for volt in range(0,4095,10):      #Iterate through possible VCPA voltage values
        dac.setVoltage(volt)          #Apply voltage to VCPA
        time.sleep(0.05)              #Let it settle
        vol.append(volt)              #Add voltage code to list
        # Read channel 0 in single-ended mode using the settings above
        light.append(adc.readADCSingleEnded(0, gain, sps)) # Measure light level
        #time.sleep(0.15)              #Wait before next step
    #Leave loop with LCPand LED off
    GPIO.output(calLEDpin,False)       #Turn calibration LED off
    dac.setVoltage(0)                  #Turn LCP off

    #
    plt.plot(vol,light)                #Plot VCPA transmission as function of voltage
    plt.xlabel('D/A voltage code')
    plt.ylabel('Light transmission A/D counts')
    plt.title('Light transmission through VCPA')
    plt.show()                         #Show the plot

    light_index_max=light.index(max(light)) #Find index of maximum transmission
    maxVCPAvolt=vol[light_index_max]       #Find DAC voltage code for maximum transmission
    print "max DAC code", maxVCPAvolt
    return  maxVCPAvolt
voltVCPA45=cal()
voltVCPA90=0
voltVCPA0=3000

#Raspberry Pi Camera Initialization
#----------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (640, 480)
#camera.resolution = (1280,720)
camera.framerate=80
rawCapture = PiRGBArray(camera)
camera.led=False
```

```
#Auto-Exposure Lock
#------------------
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain

#Initialize flags
loop=True  #Initial state of loop flag
first=True #Flag to skip display during first loop
video=True #Use video port?  Video is faster, but image quality is significantly
        #lower than using still-image capture

while loop:
   #grab an image from the camera at 0 degrees
   dac.setVoltage(0)
   dac.setVoltage(voltVCPA0)
   GPIO.output(dc0degpin,True)
   GPIO.output(dc45degpin,False)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image0 = rawCapture.array
   # Select one of the two methods of color to grayscale conversion:
   # Blue channel gives better polarization information because wavelengt
   # range is limited
   # True grayscale conversion gives better gray balance for non-polarized light
   R=image0[:,:,1]  #Use blue channel
   #R=cv2.cvtColor(image0,cv2.COLOR_BGR2GRAY) #True grayscale conversion

   #grab an image from the camera at 45 degrees
   dac.setVoltage(0)
   dac.setVoltage(voltVCPA45)
   GPIO.output(dc0degpin,False)
   GPIO.output(dc45degpin,True)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image45 = rawCapture.array  #Capture image at 45 degrees rotation
   # Select one of the two methods of color to grayscale conversion:
   # Blue channel gives better polarization information because wavelengt
   # range is limited
   # True grayscale conversion gives better gray balance for non-polarized light
```

```
    G=image45[:,:,1]
    #G=cv2.cvtColor(image45,cv2.COLOR_BGR2GRAY)

    #grab an image from the camera at 90 degrees
    dac.setVoltage(0)
    dac.setVoltage(voltVCPA90)
    GPIO.output(dc0degpin,False)
    GPIO.output(dc45degpin,False)
    time.sleep(0.05)
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    # Select one of the two methods of color to grayscale conversion:
    # Blue channel gives better polarization information because wavelengt
    # range is limited
    # True grayscale conversion gives better gray balance for non-polarized light
    image90 = rawCapture.array  #Capture image at 90 degree rotation
    B=image90[:,:,1]
    #B=cv2.cvtColor(image90,cv2.COLOR_BGR2GRAY)
    imageDOLPi=cv2.merge([B,G,R])
    cv2.imshow("Image_DOLPi",cv2.resize(imageDOLPi,(320,240),interpolation=cv2.INTER_AREA))
#Display DOLP image
    GPIO.output(dc0degpin,False)
    GPIO.output(dc45degpin,False)
    k = cv2.waitKey(1)  #Check keyboard for input
    if k == ord('x'):   # wait for x key to exit
        loop=False

#   Prepare to leave
cv2.imwrite("image0.jpg",image0)
cv2.imwrite("image90.jpg",image90)
cv2.imwrite("image45.jpg",image45)
cv2.imwrite("RGBpol.jpg",cv2.merge([B,G,R]))
cv2.imwrite("image0g.jpg",R)
cv2.imwrite("image90g.jpg",G)
cv2.imwrite("image45g.jpg",B)
dac.setVoltage(0)                    #Turn frontLCD OFF
cv2.destroyAllWindows()
quit
```

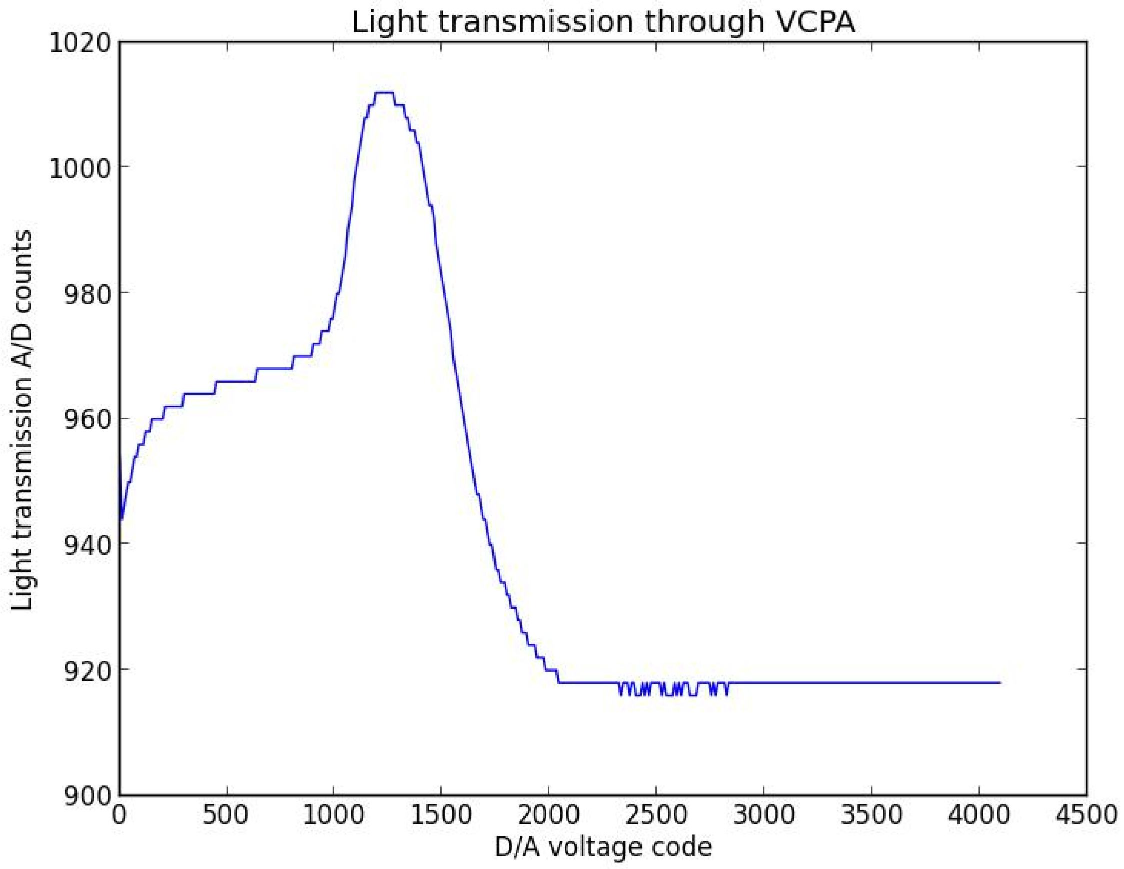


Figure 30 – The calibration procedure searches for the drive point of maximum light transmission through the VCPA illuminated by light polarized at 45^o. The x-axis is the RMS VCPA drive voltage (in DAC counts).

Please take a look at Figure 32. It shows the images that would be obtained with the VCPA set at 0^o, 45^o, and 90^o when taking pictures of a fan of polarizers at 0^o, 22.5^o, 45^o, 67.5^o, 90^o, 112.5^o, 135^o, 157.5^o, and 180^o. A single color image that encodes polarization – like that shown in Figure 33 - can be obtained with ease by assigning the 0^o, 45^o, and 90^o images to a channel of an RGB image array. You can use the Matlab function StokesParams.m in Appendix I to give this concept a hands-on try. Figure 34 shows images obtained using DOLPi for image acquisition and the Python code shown in this paper.

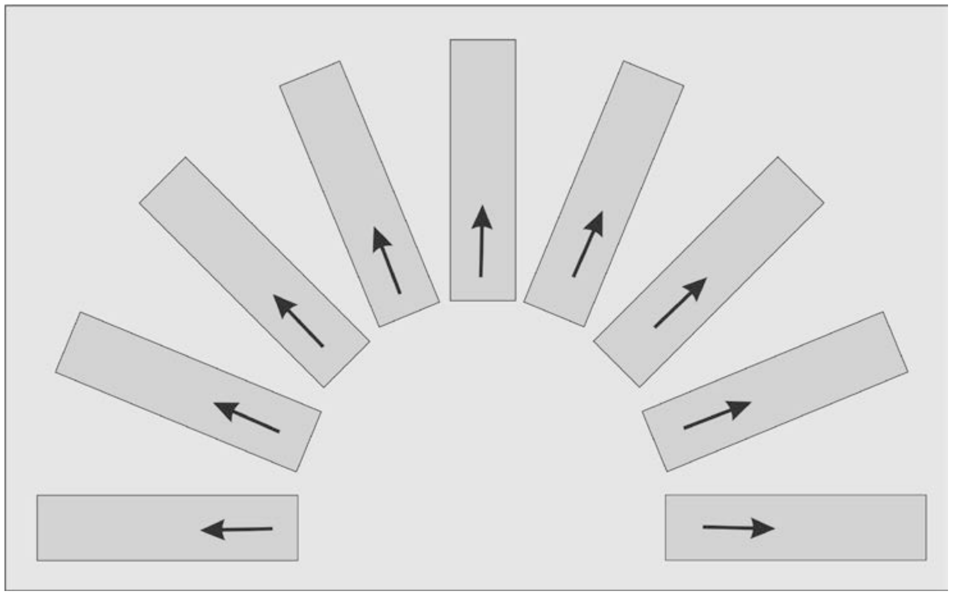


Figure 31 - A fan of linear polarizers oriented at 0°, 22.5°, 45°, 67.5°, 90°, 112.5°, 135°, 157.5°, and 180° all look the same when observed through an unaided human eye. We wouldn't even know there is something special about these rectangles if not for the arrows that I overlaid to explain the setup.

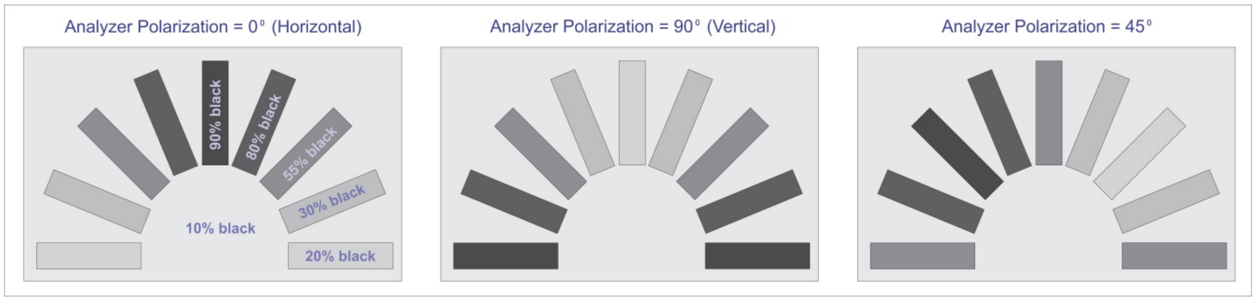

Analyzer Polarization = 0° (Horizontal)
90% black
80% black
55% black
30% black
10% black
20% black
Analyzer Polarization = 90° (Vertical)
Analyzer Polarization = 45°

Figure 32 – I synthesized these grayscale test images using CorelDraw® to test DOLPi's algorithms. The rectangles represent non-ideal polarizers at 0°, 22.5°, 45°, 67.5°, 90°, 112.5°, 135°, 157.5°, and 180°.

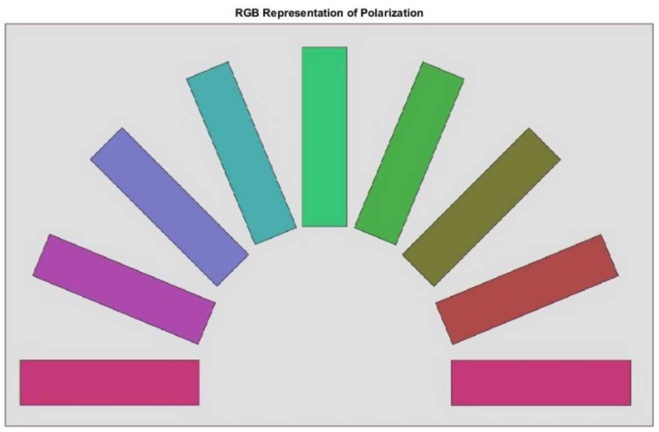


Figure 33 – A single color image that encodes polarization can be obtained with ease by combining the images acquired with the polarization analyzer set at 0°, 45°, and 90° into an RGB image. This image was obtained by combining the sample images of Figure 32. Compare this figure to the unaided view of Figure 31 – The difference is dramatic to say the least!!!

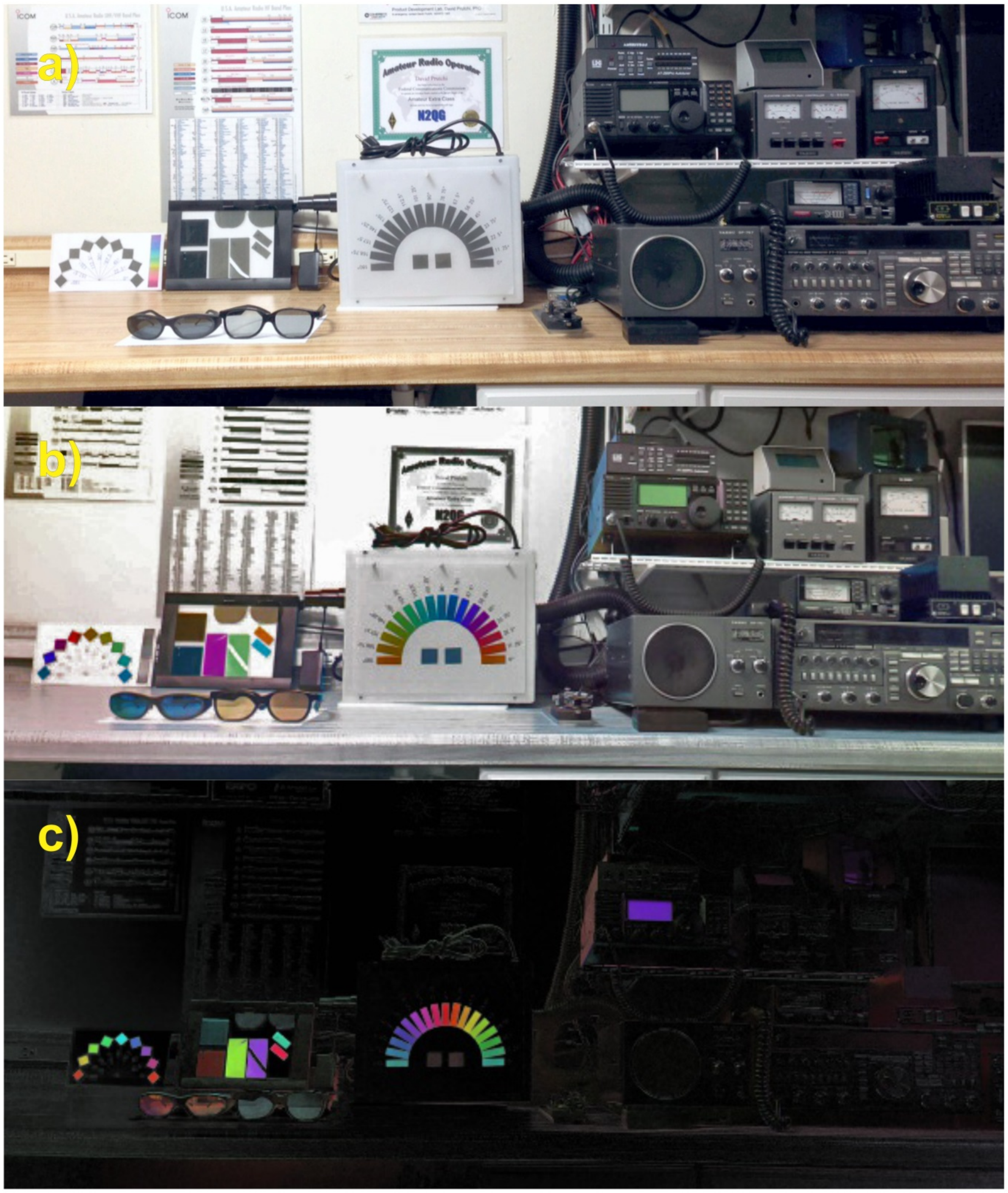


Figure 34 – This figure illustrates polarimetric imaging with some real images. Here, I placed some polarization targets (none are back-illuminated) next to my ham radio station. Nothing specific can be told about the polarizers from the visible image in (a). However, the RGB image (b) adds color to encode the angle of polarization of polarized light, while removes all color from non-polarized light (unpolarized reflections show in grayscale). The HSV image (c) goes one step further, showing only polarized light. The purple square to the right of the image is the reflection of the LCD in my HF transceiver.

Visualizing and Quantifying Polarization

Assigning the images captured with the VCPA set at 0^o, 45^o, and 90^o to the RGB color channels is one simple way of visualizing polarization. However, the development of practical applications for this camera requires some understanding of the way in which scientists and engineers describe the polarization of light under a more formal framework. Please keep on reading – I promise to keep this easy…

We must first understand that polarized light is seldom encountered in pure form. That is, objects in a scene rarely reflect light of perfectly polarized light. Instead, light is partially polarized to varying degrees because it contains components polarized in different directions. Four components are needed to completely describe light in terms of its polarization content. These are:

1. *I* = the overall intensity of the light
2. *Q* = the difference in intensity accepted through polarizers at 0^o and 90^o to the horizontal[10]
3. *U* = the difference in intensity accepted through polarizers at 45^o and -45^o to the horizontal
4. *V* = the fraction of circularly polarized light

As I had mentioned before, circularly polarized light is relatively rare in nature and often ignored, although there are some natural processes and animals that reflect and use circularly polarized light. Circular polarization is widely used today in 3D displays.

You may find these components described in the literature as the "Stokes parameters" $S_0=I$, $S_1=Q$, $S_2=U$, and $S_3=$ V. They are often organized in matrix form as the "Stokes vector":

$$S=\begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix}=\begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix}$$

where commonly only the first three parameters are usually measured as the three "linear Stokes parameters":

$$S_{linear}=\begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix}=\begin{pmatrix} I \\ Q \\ U \end{pmatrix}$$

Don't let the mathematical notation scare you! This is really easy to understand – each pixel in the linear linear-polarization-analyzed image has three parameters. The first is intensity, so a grayscale image of S_0 (or *I*) is simply the grayscale image that would be obtained with no polarizers. The second parameter S_1 (or *Q*) of each pixel is just the difference between the intensity of the pixel when observed

[10] Using the horizontal plane as reference is one possible convention. However, a plane at any other angle may be used as the reference plane at 0^o.

through a polarizer oriented horizontally[11] (0^o) and when observed through the oriented vertically (90^o). The third parameter is the similar to the second, but with polarizer orientations of 45^o and -45^o.

So, for example, completely unpolarized light would be represented by:

$$S_{linear\_unpolarized} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Light that is perfectly horizontally-polarized is represented by:

$$S_{linear\_horizontal} = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Light that is perfectly vertically-polarized is represented by:

$$S_{linear\_horizontal} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$$

Light that is perfectly polarized at 45^o is represented by:

$$S_{linear\_horizontal} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

And light that is perfectly polarized at 45^o is represented by:

$$S_{linear\_horizontal} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

For the sake of completeness, perfectly circularly-polarized light would be represented by the full Stokes vector as:

[11] Assuming that we have selected the horizontal plane to be the reference plane for 0^o, but we may choose any other plane to be the reference plane.

$$S_{right-hand} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad \text{or} \quad S_{left-hand} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

DOLPi measures intensity at 0^{o}, 45^{o}, and 90^{o}. Let's call these $I_{0°}$, $I_{45°}$, and $I_{90°}$, from which we need to calculate *I*, *Q*, and *U*. This would be straightforward if we would have a measurement at -45^{o} because the linear Stokes vector would be:

$$S_{linear} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} I \\ Q \\ U \end{pmatrix} = \begin{pmatrix} I_{0°} + I_{90°} \\ I_{0°} - I_{90°} \\ I_{45°} - I_{-45°} \end{pmatrix}$$

However, the measurement at -45^{o} is not necessary, because it can be derived from the intensity of the other three angles, so all of the linear Stokes parameters can be calculated from $I_{0°}$, $I_{45°}$, and $I_{90°}$:

$$S_{linear} = \begin{pmatrix} S_0 \\ S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} I \\ Q \\ U \end{pmatrix} = \begin{pmatrix} I_{0°} + I_{90°} \\ I_{0°} - I_{90°} \\ 2I_{45°} - I_{0°} - I_{90°} \end{pmatrix}$$

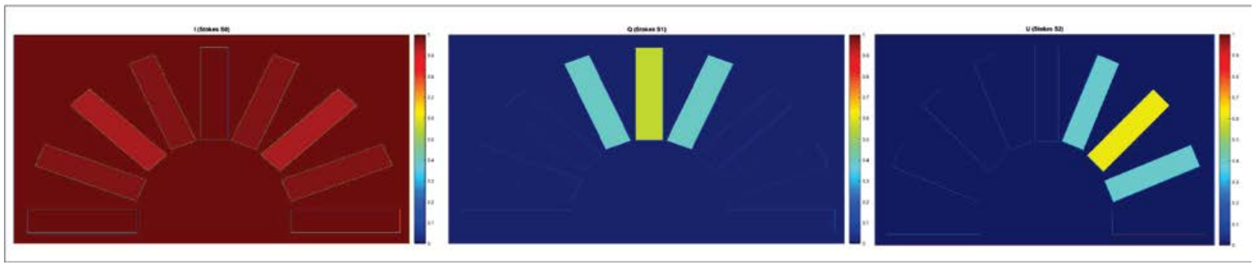


Figure 35 – The Linear Stokes parameters calculated from the images of Figure 32. As the left image (S_0=Intensity), the polarizers barely contrast with the background in my sample image.

The traditional way is to display polarization information as separate images for the intensity, degree of linear polarization, and the polarization angle. Strictly speaking, the Degree of Linear Polarization (DoLP) image is calculated for every pixel according to:

$$\text{DoLP} = \frac{\sqrt{Q^2 + U^2}}{I}$$

While the Angle of Polarization (AoP) is calculated as:

$$\text{AoP} = \frac{1}{2}\arctan\left(\frac{U}{Q}\right)$$

Overall intensity is typically displayed with a normal grayscale or color image of a scene, degree of polarization is displayed with a grayscale image of the same scene (black = 0% DoLP, white = 100% DoLP), and the angles of polarization are illustrated with different hues.

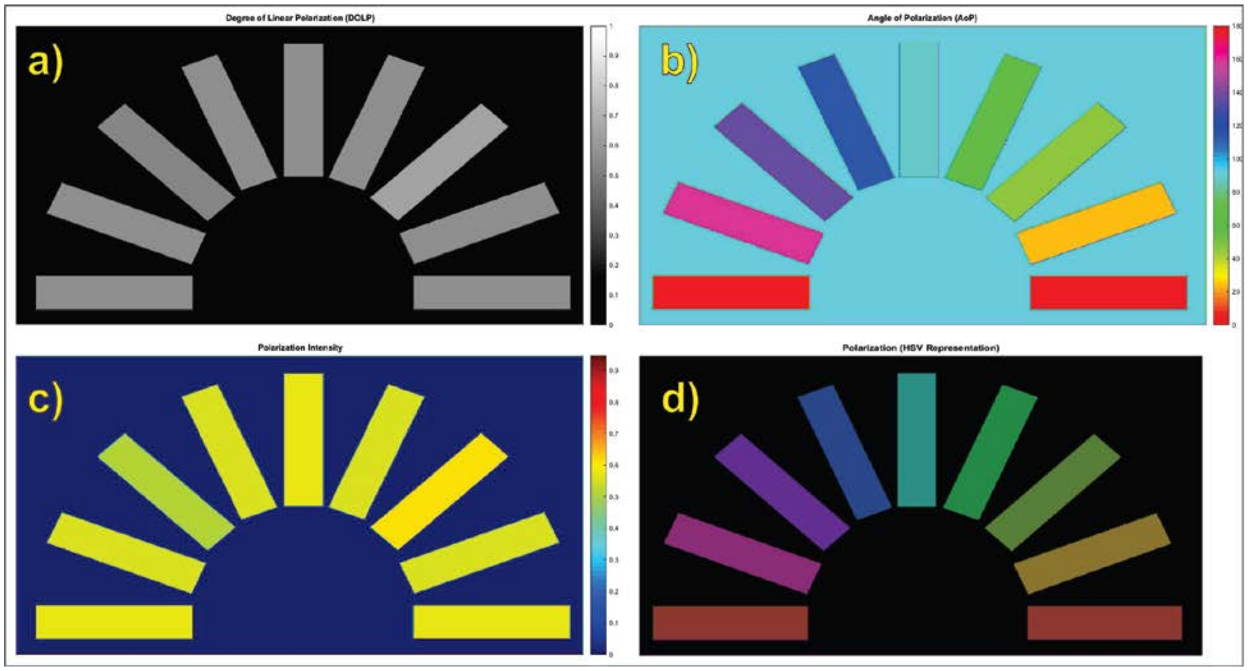


Figure 36 – The traditional way is to display polarization information as separate images for a) the Degree of Linear Polarization (DoLP), b) the angle of polarization (AoP), and c) the polarization intensity. These separate images are sometimes combined into a single color image using the Hue-Saturation-Intensity color model (d).

A popular way of presenting polarization information is by recognizing that the three main polarization components – polarization intensity, Degree of Linear Polarization, and Angle of Polarization – are analogous to the color components of brightness, saturation, and hue. The polarization parameters merged into the HSV (Hue, Saturation, Value) color space is shown in Figure 37.

In the HSV color model, as hue varies from 0 to 1, the corresponding colors vary from red through yellow, green, cyan, blue, magenta, and back to red, so that there are actually red values both at 0 and 1. To represent AoP, a hue value of 0 is assigned to 0^{o}, while a hue value of 1 is assigned to 180^{o}.

Saturation is assigned to the Degree of Linear Polarization. As DoLP varies from 0 to 1, the corresponding colors (hues) vary from unsaturated (shades of gray) to fully saturated (no white component).

Lastly, value (brightness) is assigned to Polarization Intensity. As it varies from 0 to 1.0, the corresponding colors become increasingly brighter.

The following Python code snippet implements full polarization imaging and HSV representation in DOLPi:

Table 7 – Python code snippet implementing full polarization imaging and HSV representation in DOLPi-MECH. The same can be used in the electro-optic version of DOLPi by eliminating the parameters related to circular-polarization.

```
#convert images to signed double (int16)
image0_d=np.int16(image0)
image90_d=np.int16(image90)
image45_d=np.int16(image45)
image135_d=np.int16(image135)
imageLHCP_d=np.int16(imageLHCP)
imageRHCP_d=np.int16(imageRHCP)
#calculate Stokes parameters
stokesI=image0_d+image90_d+.1 #.1 added here to prevent division by zero later on
stokesQ=image0_d-image90_d
stokesU=image45_d-image135_d
stokesV=imageLHCP_d-imageRHCP_d
#calculate polarization parameters
polInt=np.sqrt(1+np.square(.1+stokesQ)+np.square(.1+stokesU))  #Linear Polarization Intensity
                                    #.1 added because np doesn't like to find
                                    #the square of zero
polDoLP=polInt/stokesI   #Degree of Linear Polarization
polAoP=0.5*(np.arctan2(stokesU,stokesQ))  #Angle of Polarization
polDoCP=(2*np.absolute(stokesV))/polInt  #Degree of Circular Polarization

#prepare DOLPi HSV image
H=np.uint8((polAoP+(3.1416/2))*(180/3.1416))
S=np.uint8(255*(polDoLP/np.amax(polDoLP)))
V=np.uint8(255*(polInt/np.amax(polInt)))

imageDOLPiHSV=cv2.merge([H,S,V])
DOLPiHSVinBGR=cv2.cvtColor(imageDOLPiHSV,cv2.COLOR_HSV2BGR)
cv2.imshow("DOLPi_HSV",DOLPiHSVinBGR)
```

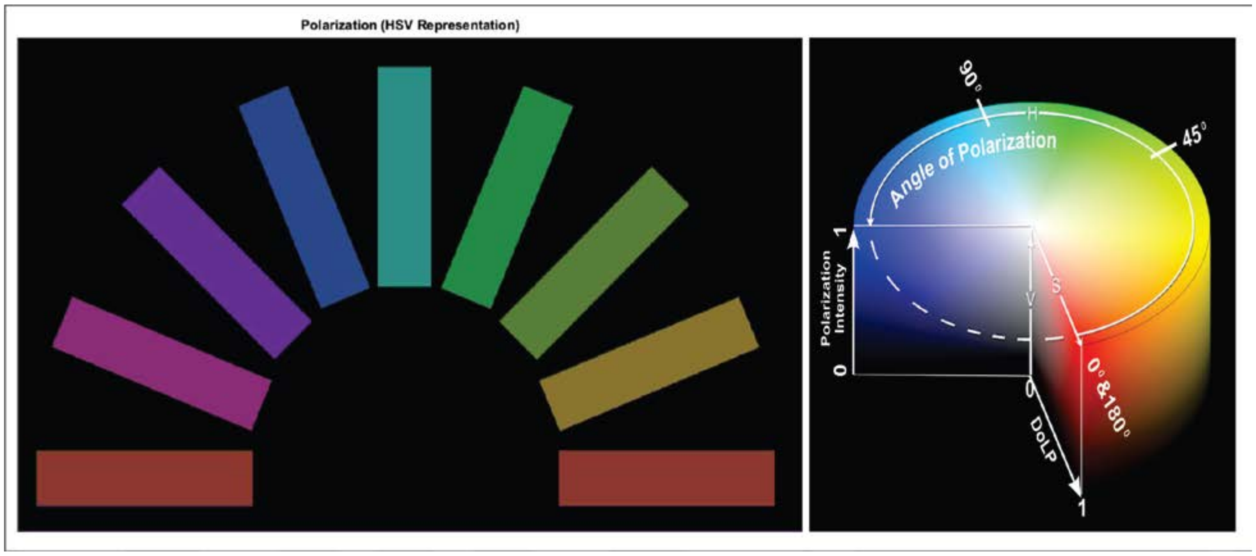


Figure 37 – HSV representation of polarization parameters (DoLP, AoP, and Polarization Intensity) for the images of Figure 32. The figure at the right is the colormap for the HSV image.

It is interesting to compare the image produced by a simple RGB merge of the individual grayscale images acquired by setting the analyzer at 0^o, 45^o, and 90^o (Figure 38-a) with that obtained from combining the calculated DoLP, AoP, and Polarization Intensity images through the HSV model (Figure 38-b). The HSV model provides better contrast and more quantifiable information about a scene's polarization. However, the calculation of intermediate parameters takes processing time, so the simple RGB merge is often a good tradeoff as a preview for the sake of attaining a higher frame rate[12]. A real-world example of an image that has been acquired and completely analyzed by DOLPi is shown in Figure 40.

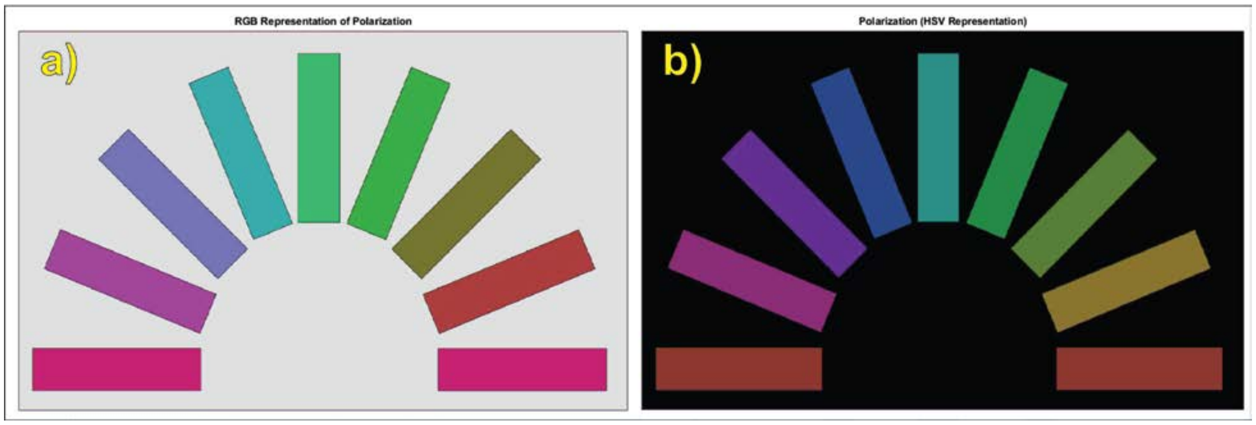

RGB Representation of Polarization
a)
Polarization (HSV Representation)
b)

Figure 38 – A simple RGB merge of the grayscale images obtained at polarization analyzer settings of 0^o, 45^o, and 90^o (a) suffices as a real-time preview of the HSV representation (b) obtained through more complex calculation of DoLP, AoP, and Polarization Intensity from the linear Stokes parameters.

[12] Or rather may be, once higher frame rates can be attained for single-frame shots by the Raspberry Pi's GPU.

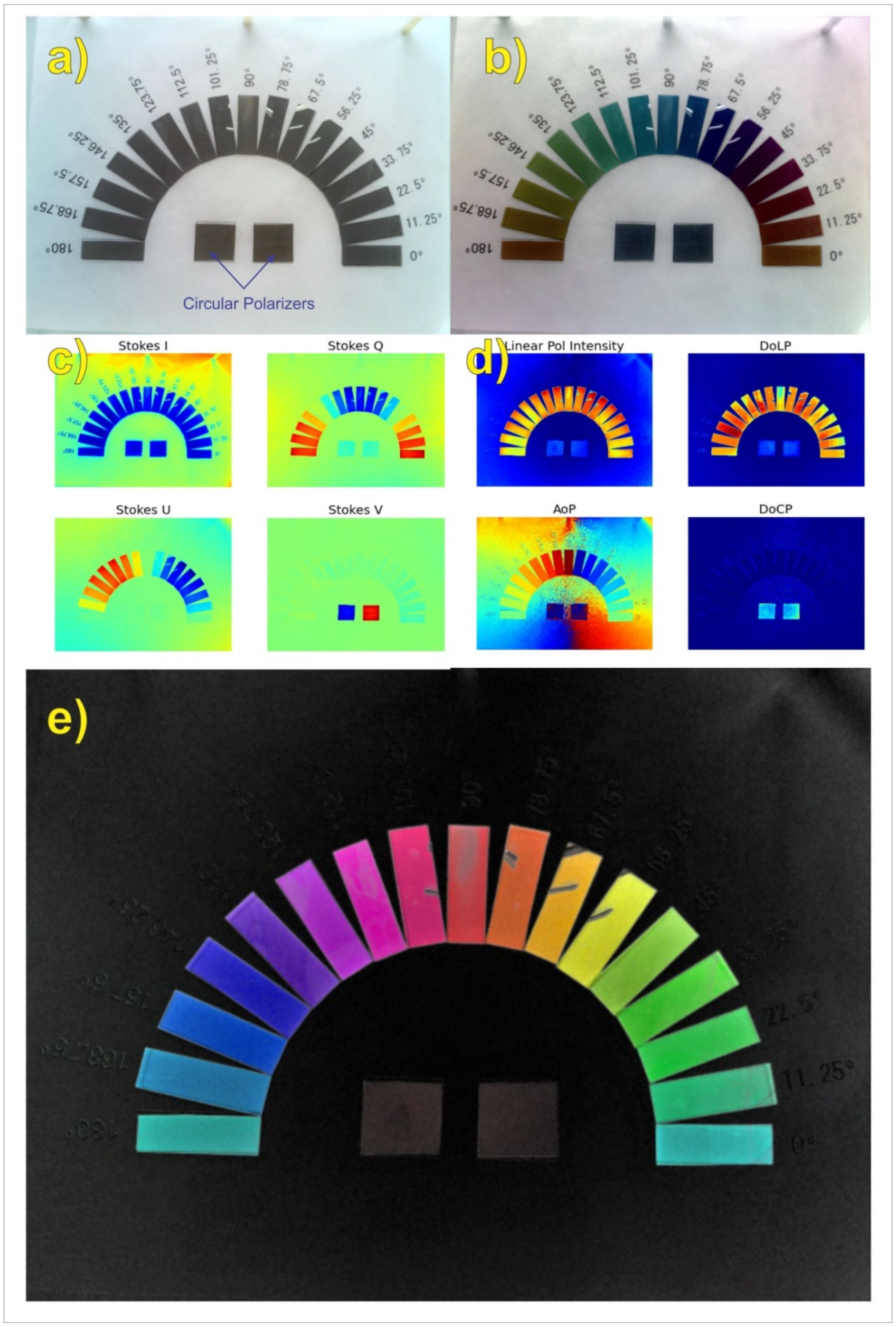


Figure 39 - Full polarimetric analysis by DOLPi-MECH. a) Normal visible image of the polarization test target. The rectangles are linear polarizers oriented as shown. The two bottom squares are made of circular polarizing film – one RHCP and one LHCP b) RGB-encoded polarization image. Colors in this image relate to the angle of linear polarization. c) Stokes parameters calculated by the DOLPi software. Note that the Stokes V parameter shows the circular polarizers. d) Polarimetric parameters calculated by the DOLPi software. e) HSV-encoded polarimetric image clearly shows linear polarized light where color (Hue) encodes AoP.

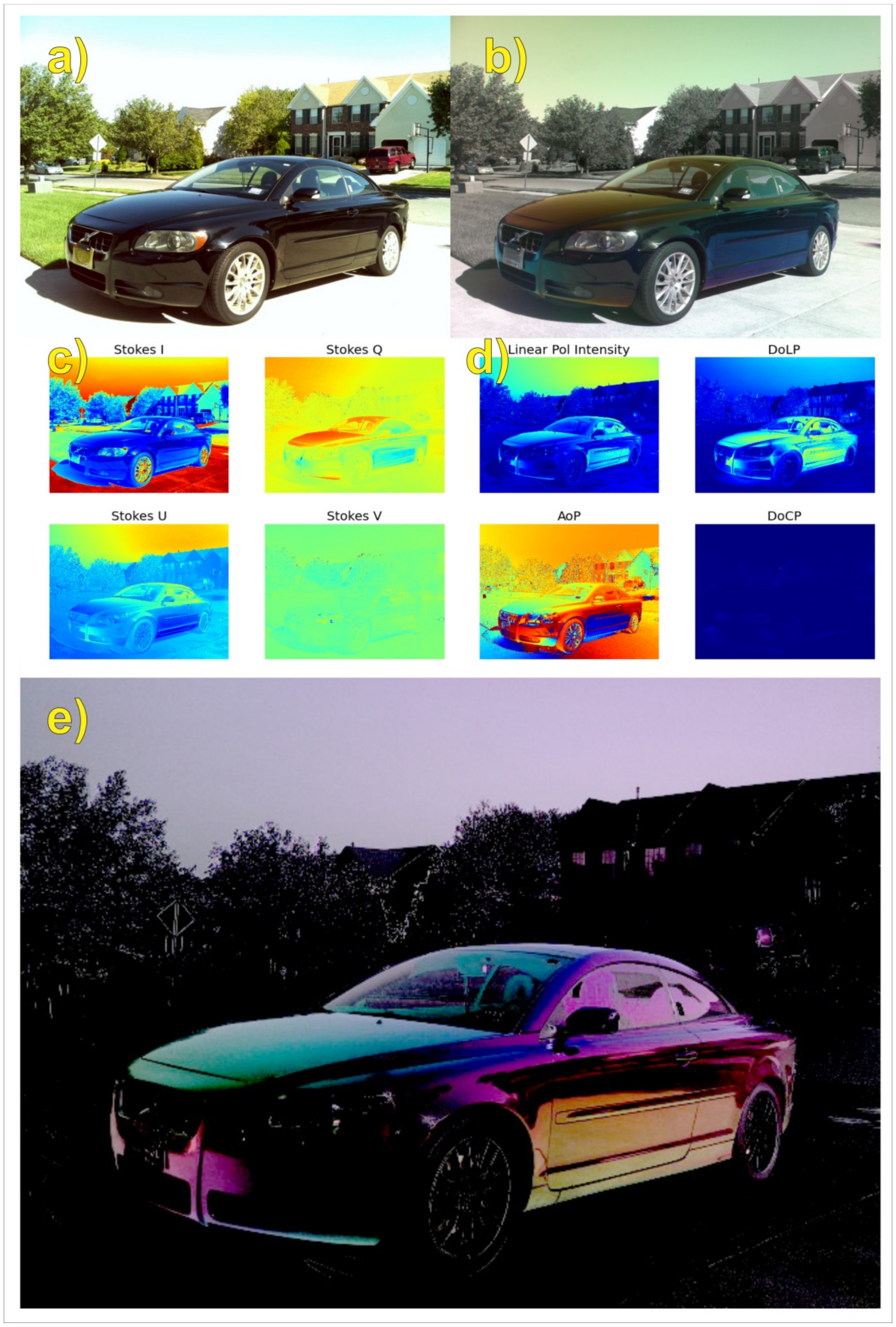


Figure 40 – Real-world polarimetric imaging using DOLPi-MECH. a) Normal visible image of the scene. b) RGB-encoded polarization image. Colors in this image are not related to the actual color (wavelength) of the light. Instead, colors relate to the angle of linear polarization. c) Stokes parameters calculated by the DOLPi software. d) Polarimetric parameters calculated by the DOLPi software. e) HSV-encoded polarimetric image clearly shows surfaces that reflect polarized light. Please note the absence of circularly-polarized light in this image, which is typical of the vast majority of outdoors scenes.

One last bit of mathematics is worth mentioning - the “Muller Matrix” represents the way an optical component affects the polarization of light. The Muller Matrix has the form:

$$\begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix}$$

As we have seen, the polarization of light is represented by Stokes vectors, but a mathematical operator is needed to describe the way in which some optically-active substance changes the polarization of light. This is the purpose of Muller calculus using the Muller matrix:

[Polarization of output light] = [Muller matrix] × [Polarization of input light]

Or, formally:

$$\begin{pmatrix} I' \\ Q' \\ U' \\ V' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{pmatrix} \times \begin{pmatrix} I \\ Q \\ U \\ V \end{pmatrix}$$

This is worth mentioning, because papers that discuss polarimetric imaging applications often refer to these Muller matrices to describe the way in which materials change the polarization of light. For example, cancerous cells shift the polarization of light differently than healthy cells, so their Muller matrices are different [Antonelli et al., 2009; Novikova et al., 2013], allowing for the development of cancer-detection methods based on polarimetric imaging.

I hope that this section didn’t give you a headache! Although the math is relatively simple, you can now appreciate that it is difficult for us to intuitively integrate the sense of light polarization into the way we normally use our vision (that is, through the analysis of intensity and color). I believe that this unintuitive nature of polarization has been responsible for slowing down the development of practical applications for polarimetric imaging. I encourage you to review this section once again so that you can understand what is it that we are viewing when looking at a polarization image.

Benchmarking DOLPi against DOLPI-Mech

We have been ignoring circularly-polarized light, but although rare in the everyday environment, circularly-polarized light has important applications, and its presence can confuse DOLPi. Please take a look again at Figure 2 – The linear polarizer strips oriented at 0^o, 45^o, and 90^o show distinct colors representative of their angle. However, note that the two round (-ish) circular polarizer film pieces (taken from RealD 3D glasses) on the top right corner also show distinct colors. A strict linear analysis should show them to have virtually no linear polarization. Using a strict linear Stokes imaging polarimeter these pieces should both show as the same grayscale level.

I had mentioned in an earlier footnote that the LCP driven half-way acts as a “quarter-wave plate”, which converts light polarized at 45º into circularly-polarized. An ideal LCP driven into this mode would become a circular-polarization analyzer rather than a linear polarization analyzer set at 45º. This is the reason why the folks at Bossa Nova Technologies used multiple LCPs and other polarization components to analyze light at 45º in their Salsa camera (Figure 20).

My first DOLPi prototype followed a multi-LCP architecture, and I was thus very surprised while testing the LCP that was meant to act as a variable retarder, when I saw that the polarization images that I was obtaining with just this one LCP (driven at 3 states) were qualitatively better than those using the full stack[13]. Today, based on my experiments, I’m convinced that DOLPi’s “45 degree image” indeed contains a dominant 45 degree component when observing linearly polarized light.

Searching for a good explanation about the processes that make this possible, I came across a paper published in 2008 [Jaulin et al., 2008 and its accompanying presentation by Bigué et al.] in which researchers from the Université de Haute-Alsace in France described the implementation of an imaging polarimeter using a single liquid crystal light modulator. The math is complex, so I’ll spare you, but the paper showed that a full linear Stokes polarization analyzer (taking pictures at 0º, 45º, and 90º) can be made with a single liquid-crystal cell.

As shown in Figure 41, the electro-optic polarization analyzer in DOLPi cannot differentiate 45º linearly-polarized light from that with circular polarization. This is one of the electro-optic DOLPi’s shortcomings. However, given the rarity of circularly-polarized light in nature, this drawback is not important for the vast majority of possible applications. In fact, for real-world applications, qualitative information of polarization contrast may be more useful and important than a precise quantitative polarimetric analysis.

[13] This is fortunate, since the polarimetric imaging technique used in the Bossa Nova Salsa camera is patented [Lefaudeux, 2011].

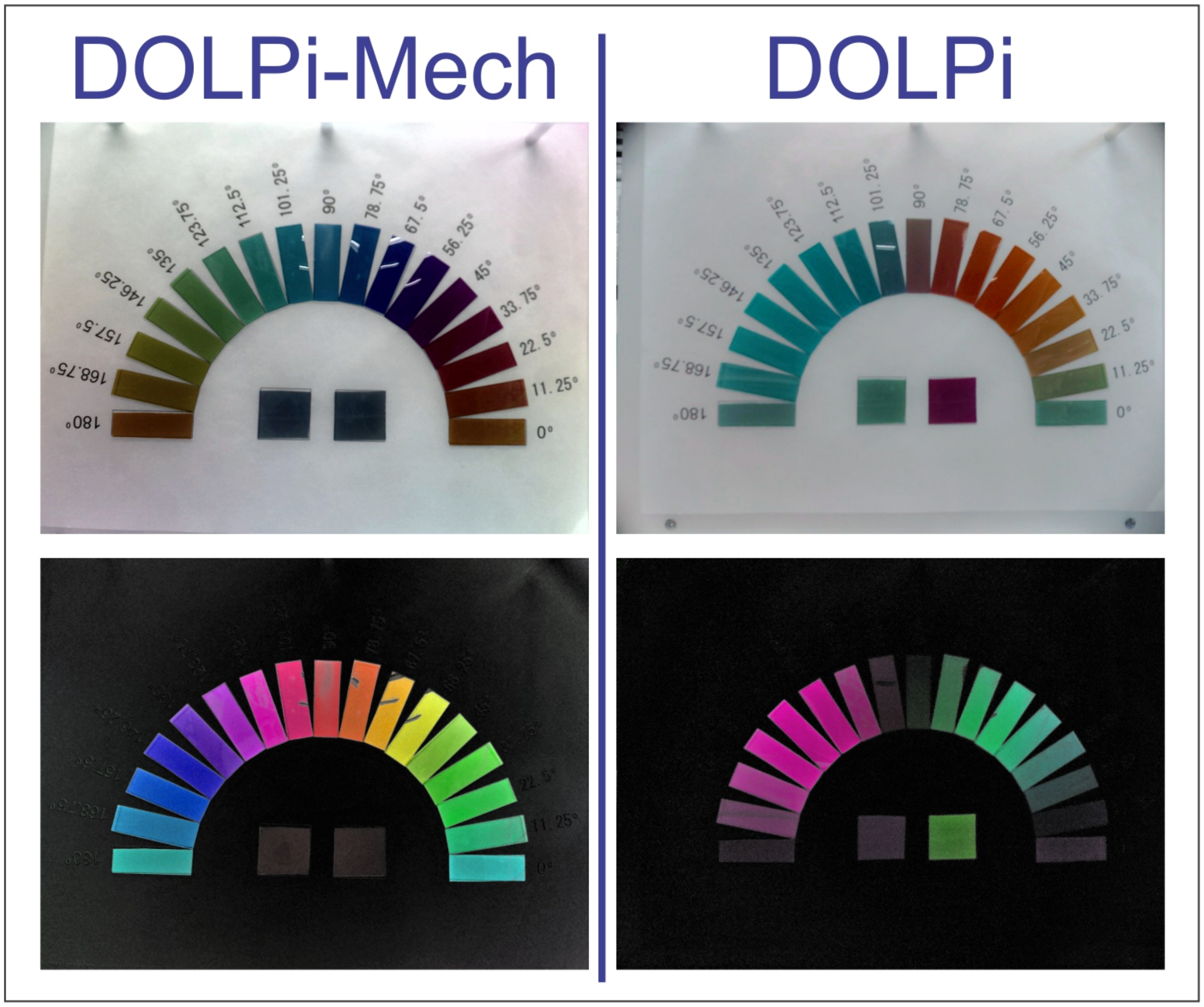


Figure 41 – DOLPi's LCP driven half-way acts as a "quarter-wave plate", which converts light polarized at 45° into circularly-polarized. An ideal LCP driven into this mode would become a circular-polarization analyzer rather than a linear polarization analyzer set at 45°. Using a strict linear Stokes imaging polarimeter such as DOLPi-MECH shows the two circularly-polarized squares on the bottom of the target as the same grayscale level (left hand figures). However, the electro-optic polarization analyzer in DOLPi cannot differentiate 45° linearly-polarized light from that with circular polarization. This is one of the electro-optic DOLPi's shortcomings. However, given the rarity of circularly-polarized light in nature, this drawback is not important for the vast majority of possible applications.

Polarized Light in the Environment

Non-polarized light becomes polarized when it is reflected by a nonmetallic surface. The degree of polarization depends on the angle at which the light hits the surface, as well as on the type of reflective material. As shown in Figure 42 and Figure 43, water, snow fields, and asphalt roads reflect the Sun's light with very strong horizontal polarization (i.e. parallel to the reflective surface).

The glare produced by these reflections can be diminished through the use of vertically-polarized sunglasses. Next time you go to a fishpond, pay attention how much easier it is to see fish when your sunglasses are in their "normal" position (the orientation of their polarizers being vertical) versus when you tilt your head sideways and thus align your sunglasses' polarization with that of the water's surface.

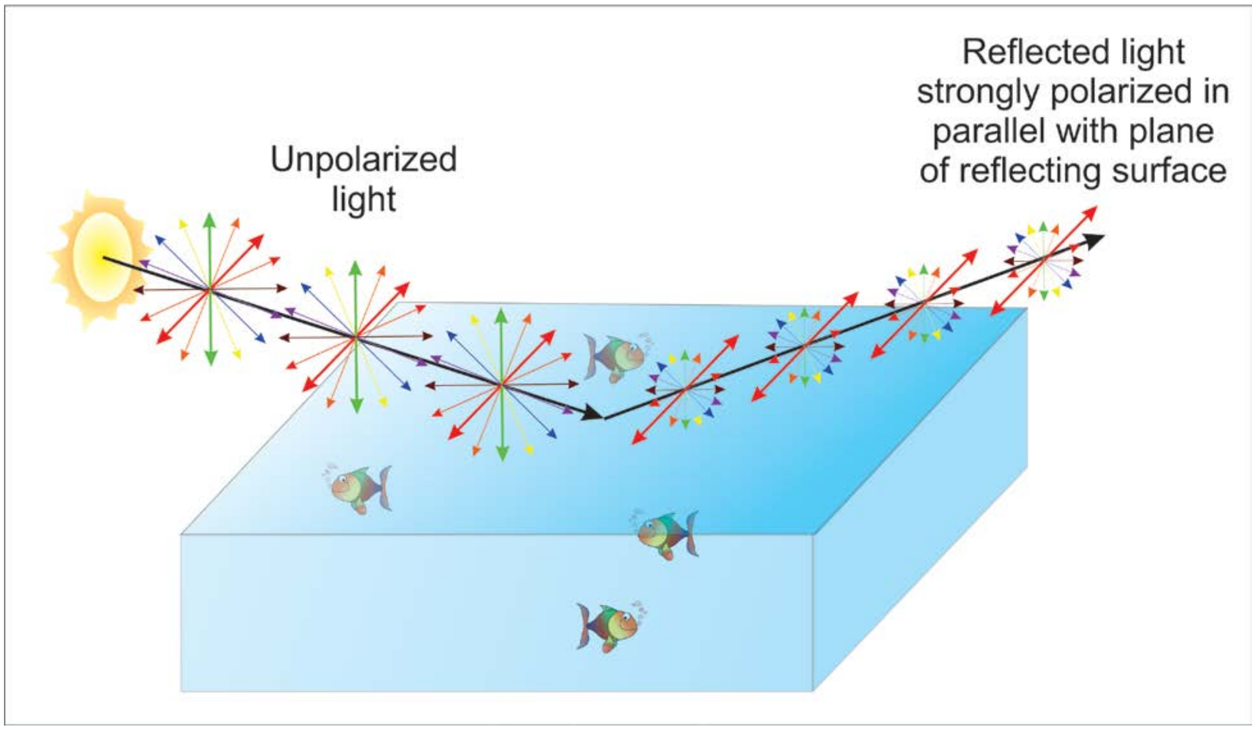

Unpolarized light
Reflected light strongly polarized in parallel with plane of reflecting surface

Figure 42 – Non-polarized light becomes polarized when it is reflected by a nonmetallic surface. The degree of polarization depends on the angle at which the light hits the surface, as well as on the type of reflective material. Water, snow fields, and asphalt roads reflect the Sun's light with very strong horizontal polarization. Vertically-polarized sunglasses thus cut the glare produced by horizontally-polarized reflections.

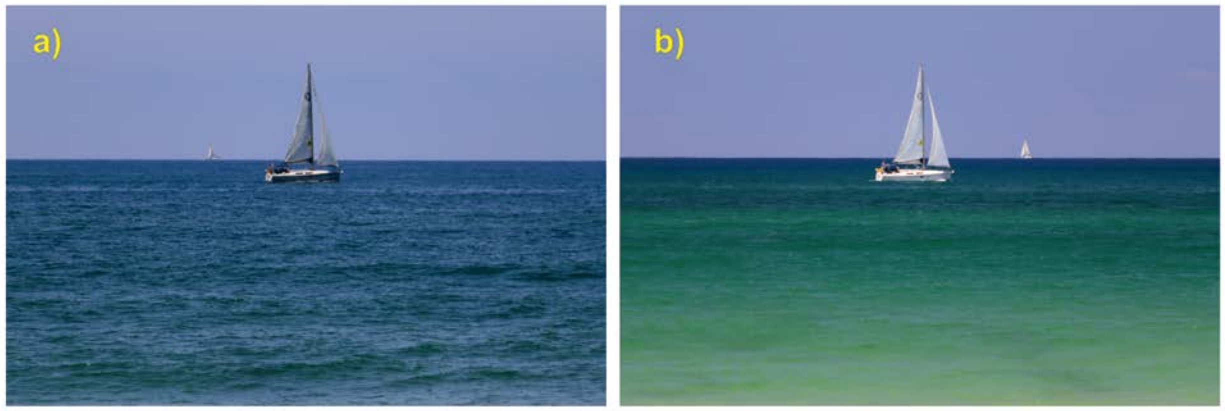


Figure 43 – Coastal scene viewed through a polarizer. a) Horizontal polarizer filter allows light reflected by the water surface to reach the camera; while (b) vertical polarizer cuts the strong horizontally-polarized reflection, allowing underwater features to be captured. The boats and sails, made of plastic materials also polarize light very strongly (in this case vertically – the opposite of the water and sky), making them contrast very strongly in the image obtained through the vertical polarizer. (Images shot from Hof HaTzuk, Israel with Canon 6D and Hoya CPL).

Many aquatic insects use their polarization-sensitive vision to locate ponds. For example, insects that lay eggs in water find and select egg-laying sites based on the intensity of polarization from reflective water surfaces. Unfortunately, many human-made objects can reflect horizontally polarized light so strongly that they appear to aquatic insects to be bodies of water. Some – like solar panels – reflect light with such high level of horizontal polarization (close to 100%) that they look to aquatic insects as far superior breeding grounds than pond water (which reflects light with a degree of polarization of around 30 to 70%), becoming ecological death traps to these organisms [Horvath et al, 2010]. As a consequence, or maybe even through the same polarization-based location mechanism, bird populations may be affected by solar farms because the disruption of a bird's natural patterns of behavior may lead to disorientation and increased energy use [BirdLife, 2015].

NASA has started to mimic insect vision and experiment with polarimetric imaging to locate ponds from satellites and high-flying aircraft. You may think that this is a trivial problem, but in reality classical multispectral imaging often confuses shallow, oxygen-rich ponds with vegetation fields. This is because such ponds serve as good breeding grounds for algae that make their surface green. However, as shown in Figure 44, the water surface reflects highly polarized light even in the presence of the algae.

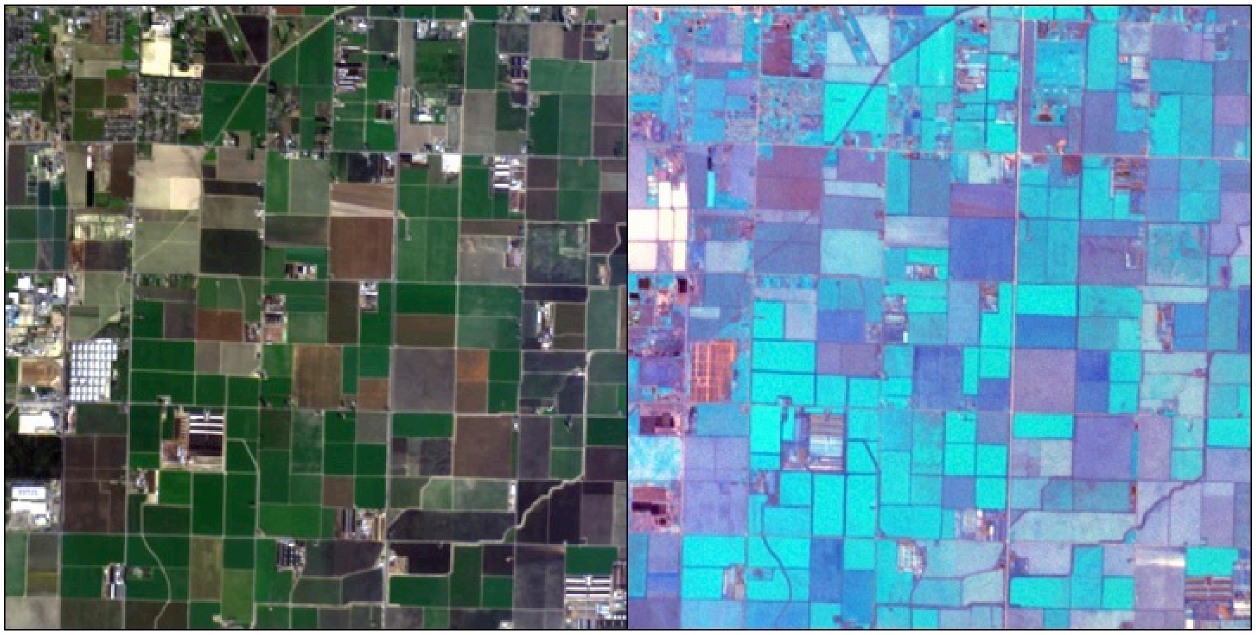


Figure 44 – Images obtained by the Jet Propulsion Laboratory's Airborne Multispectral Polarimetric Imager (AirMSPI) from NASA's ER-2 high altitude aircraft flying over the California Central Valley. Left panel shows a normal visible-range image, while the panel on the right shows DoLP at 470, 660, and 865 nm wavelengths. The most prominent DoLP features are the bright white regions in the far center left of the image. While these look like fields in the true color image, they are actually holding ponds for a wastewater treatment plant.
Image credit: JPL, https://eosweb.larc.nasa.gov/project/airmspi/images/AirMSPI_Hanford_Image_Pair_286F[1].jpg

Another natural source of polarized light is scattering of sunlight in the atmosphere. The exact mechanism is outside the scope of this article, but suffice it to say that sunlight that is scattered backwards (or forward) by the atmosphere remains unpolarized, while light scattered at 90^{o} degrees from the Sun's position becomes linearly polarized, while light scattered at intermediate angles is only partially polarized (Figure 45). If you are interested in more detail, Glenn S. Smith from Georgia Tech published an excellent, easy-to-understand paper with college-level math that really explains the process of sunlight polarization by atmospheric scattering [Smith, 2007].

As shown in Figure 25, the polarization vectors in the sky are all oriented along parallel circles centered on the Sun's position. Being able to see the distribution of polarization angles can be used as an orientation cue. In fact, sky polarization patterns are used by many insects for navigation. For example, honeybees use celestial polarization to move between the hive and foraging locations. Salmon are thought to have similar capabilities to orient themselves based on the sky's polarization as seen underwater.

It is hypothesized that the Vikings knew how to navigate this way based on a legend that tells of a glowing "sunstone" that, when held up to the sky, revealed the position of the Sun even on a cloudy day. This would have been important since constant daylight during the summer sailing season at high latitudes would have prevented them from using the stars as a guide to their positions. Even a magnetic compass, which is believed to have been unknown to the Vikings, would have been almost useless so close to the magnetic North Pole. The theory thus goes that a natural polarizing crystal such as calcite would have helped the Vikings find the position of the Sun even through thick fog and overcast skies [Horváth, 2011].

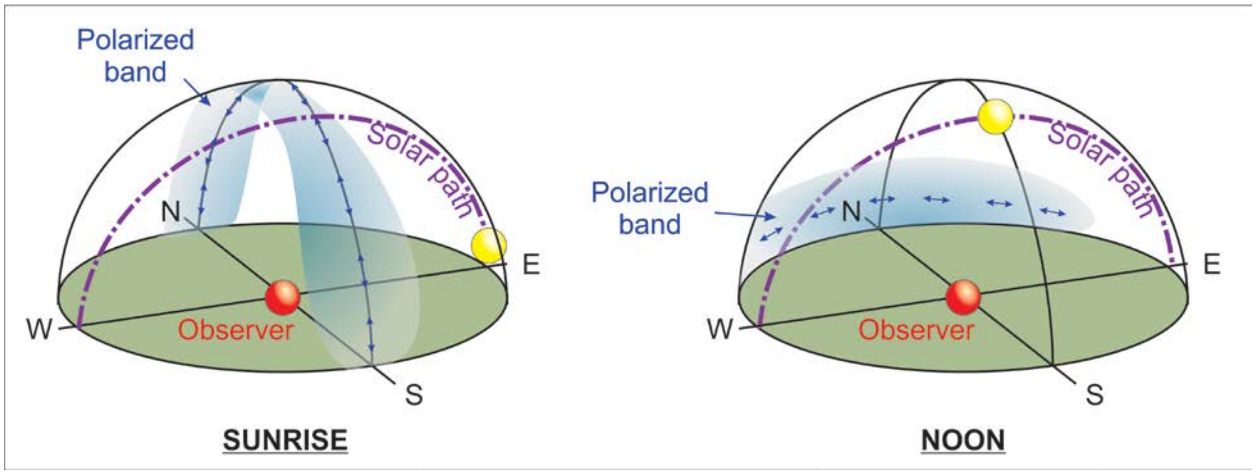


Figure 45 – The sky is polarized tangential to a circle centered on the Sun. A band of maximum polarization occurs 90^o from the Sun. At sunrise and sunset, the sky is maximally polarized along the meridian, and thus vertically-polarized at the horizon. On the other hand, at noon the band of maximum polarization is horizontally polarized along the horizon. Many insects use sky polarization patterns for navigation. Honeybees use celestial polarization to move between the hive and foraging locations.

So… What about the Cloaked UFOs?

I have never seen a UFO, and don't know anyone who claims to have seen one either. In addition, I am not aware of any compelling evidence that we are being visited by extraterrestrial beings, so I remain skeptical. However, it is fun to imagine how massive alien (or military) craft like the ones shown in Figure 46 could be made to fly in our skies without being seen. Maybe the craft could be rendered transparent like Wonder Woman's airplane? Maybe it could bend or reflect the light around it so that it is cloaked within a bubble of invisibility? Maybe it could use an active cloaking screen that paints an image of the background on its surface[14]?

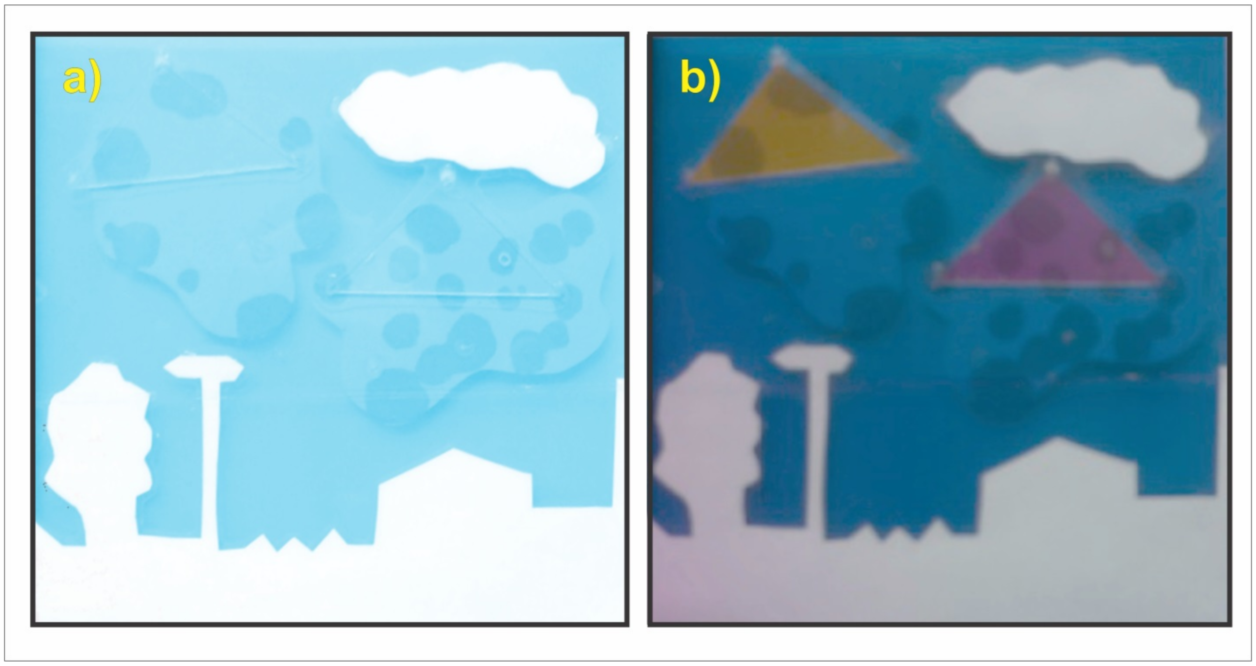


Figure 46 – It's fun to think how UFOs and secret military aircraft could be flying cloaked right above us without being noticed (a), but polarimetric imaging could possibly work against artificial cloaking devices because there is almost always a polarization change whenever light interacts with matter (b). The polarizing sky is simulated by a linear polarizer sheet, while the triangular "UFOs" are cut from transparent retarder films. The image on the left was snapped with my iPad, while the one on the right is the DOLPi RGB preview image.

We don't have to go extraterrestrial to find examples of these cloaking methods. Just go into the ocean and you'll find dozens of species that camouflage themselves through transparency and mirror reflection. However, these cloaking strategies can be defeated by animals that have evolved polarization-sensitive vision, and the same could possibly work against artificial cloaking devices because there is almost always a polarization change whenever light interacts with matter. As a matter of fact, polarization cues are so important underwater that evolution has given some animals – like cuttlefish (Figure 47) – polarization-sensitive instead of color-sensitive vision [Les, 2012].

[14] This last one is actually possible with today's technology - Electro-optical camouflage, also known as "adaptive camouflage", is described in U.S. Patent Number 5,307,162 to Schowengerdt [1994]). The cloaking system uses a thin video screen between the observer and the object being concealed while at the same time presenting a full color image of the background on the screen, thus providing the illusion that the object is not there.

Figure 47 – Cuttlefish are color-blind, but have polarization-sensitive vision with a resolution of about 1^o. They are often referred to as the "chameleons of the sea" because of their remarkable ability to rapidly alter their skin color and pattern - including polarization - to camouflage themselves, to scare potential predators, and to communicate with other cuttlefish. Image credit: "Cuttlefish @ Oceanário de Lisboa" by David Sim via Wikimedia Commons.

Transparent animals are never completely transparent, but their refraction index is sufficiently well matched to that of water that it is difficult to see from a distance. However, transparent tissues modify the polarization of light, either by increasing or decreasing its polarization state, such that they will contrast against their background when seen through polarization-sensitive eyes. For example, polarization-sensitive squid can detect zooplankton prey at 70% greater distance under partially polarized lighting than under nonpolarized lighting.

In the same way, cancerous cells reflect and scatter light with different polarization characteristics than those of healthy cells. Just as with color, polarization differences can provide additional contrast cues to detect malignant cells. This technique is being investigated by a number of researchers [Antonelli et al., 2009, Manikova et al, 2013], and I believe that DOLPi could enable the development of a cheap and effective "smart mirror" for at-home skin cancer screening.

Along similar lines, polarimetry has been shown to provide advantages in early detection of structural changes in the eye's retina, which would permit the development of non-contact techniques for early diagnosis of glaucoma, which is a leading cause for visual impairment [DeHoog et al., 2009; Wang et al., 2015].

Moving into industrial applications, polarimetric imaging can be used for on-line quality control in the manufacture of transparent objects. This is because birefringence - the optical property of a material having a refractive index that depends on the polarization and propagation direction of light – often appears in areas where the crystal structure of a transparent object is placed under mechanical stress. The level of birefringence is linearly proportional to the stress in the material, so if the material's characteristics are known, the birefringence can be directly used to calculate stress expressed in units of mechanical pressure (e.g. Pascals, psi, etc.)

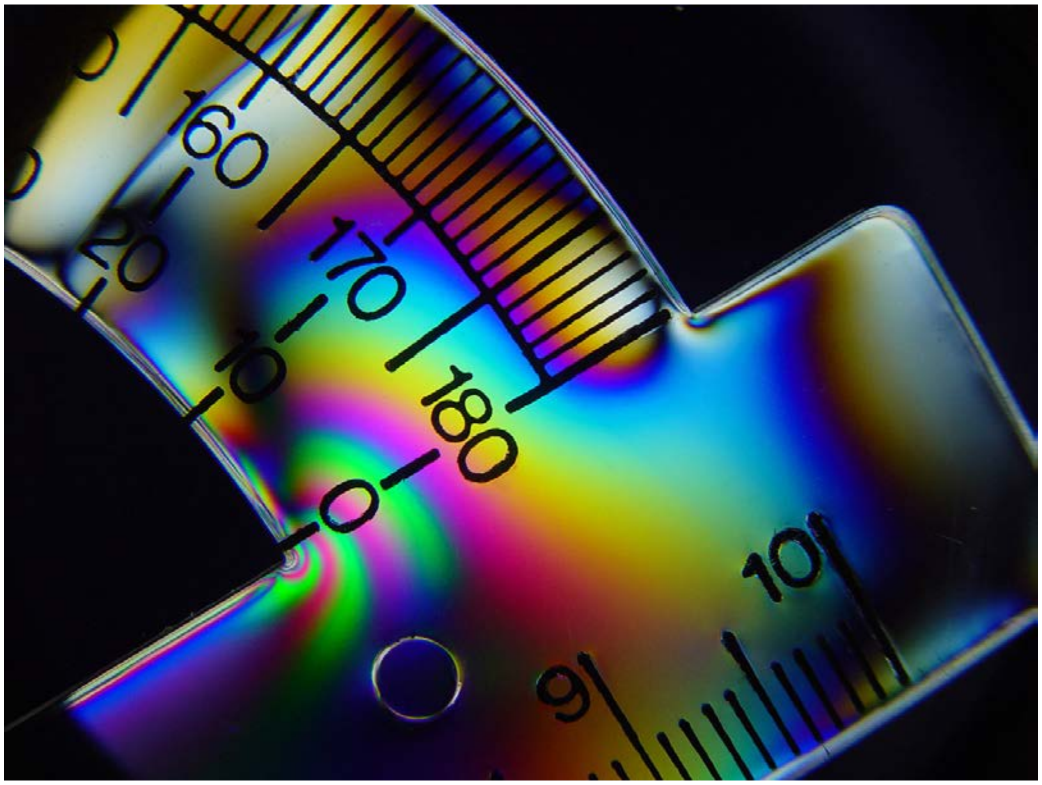


Figure 48 - A plastic protractor observed between two crossed polarizers shows areas of high residual stress concentration as bands of color. Skilled interpretation is needed to understand the stress levels represented by these colors. Image credit: "Plastic Protractor Polarized" by Nevit Dilmen via Wikimedia Commons.

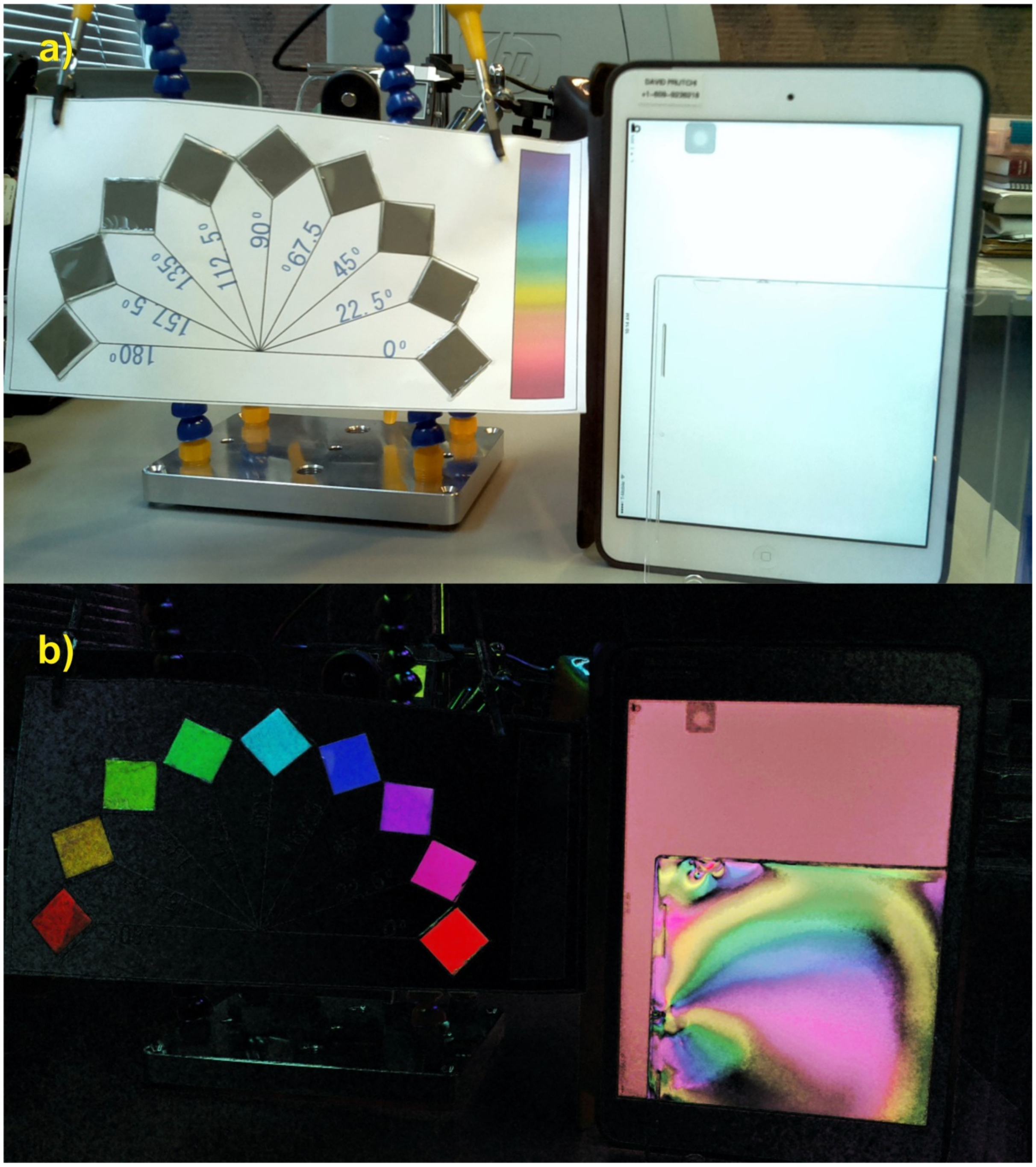


Figure 49 – DOLPi provides precise information on the polarization shifts produced by the materials, allowing for a quantitative assessment of residual stresses. a) The transparent plastic cover of a DVD case is back-illuminated with polarized light from a blank iPad Mini screen (using the Light Box - Illuminator Viewer app by Systemiko Inc). b) The HSV representation yields accurate polarization data needed to calculate stresses expressed in units of mechanical pressure (e.g. Pascals, psi). This technique can be used for real-time quality-control in the manufacture of plate glass, pharmaceutical bottles, etc.

Stress birefringence in transparent materials is caused by applied external pressure, manufacturing processes, handling and impact damage (e.g. chips, cracks, and scratches), temperature differentials, and others. Very high birefringence can often be observed with the naked eye, but more commonly, stresses in materials can be observed when a sample is placed between two crossed polarizers and a backlight.   Figure 48 shows a plastic protractor photographed between two crossed polarizers. Color bands can be observed that are proportional to the magnitude of residual stresses in the plastic. Results are subjective because they require skilled interpretation of the color shifts and birefringence. On the other hand, as shown in Figure 49, measuring the actual shift in polarization with a polarimetric camera like DOLPi yields quantifiable stress information, which can be used not only for engineering analyses, but also for real-time quality control of glass containers, plate glass, pharmaceutical bottles, and other transparent products that must tolerate external pressures without breaking.

Polarimetric imaging is an excellent contrast-enhancing technique because its information supplements the knowledge that can be derived from intensity and color imaging. Spectral characteristics (color) depend largely on the materials that make up the objects in a scene. However, the polarization properties depend strongly on the microscopic shape of a reflective surface (roughness) and its orientation. As such, spectral and polarization information supplement each other by contributing independent features for the detection and characterization of camouflaged objects or those hidden in a cluttered environment. Furthermore, man-made objects often reflect light with some degree of polarization when they incorporate non-metallic surfaces such as glass, plastic, paints, and rubbers. Since most natural backgrounds do not polarize light, man-made materials are highlighted by color in DOLPi images. Neither camouflage nor moderate foliage can escape a polarimetric imager, making it possible to develop new methods for humanitarian detection and clearance of land mines [de Jong and Schavemaker, 2007; El-Saba et al, 2008].

Drs. Wim de Jong and John Schavemaker from TNO in The Netherlands demonstrated the use of a polarimetric camera for the detection of land mines. Interestingly, the camera they used was of the rotating-filter type with performance comparable to those of DOLPi and DOLPi-MECH. Figure 50 shows results of polarimetric imaging of a cluttered test area where inert landmines had been placed. de Jong and Schavemaker [2007] concluded that a combination of intensity and polarization imaging are superior to visual-range imaging alone for an automatic landmine detection system.

I believe that landmine detection for humanitarian demining is one of the best and most promising applications of polarimetric imaging with DOLPi, and one that can save countless young lives in countries that have been ravaged by war.

Polarimetric imaging with DOLPi can thus be used for a large number of image enhancement tasks such as glare reduction, haze removal, background removal, and contrast enhancement. These are highly valuable for forensic work, industrial and environmental monitoring, microscopic investigation of tissues and plastics, and autonomous vehicle vision just to name a few.

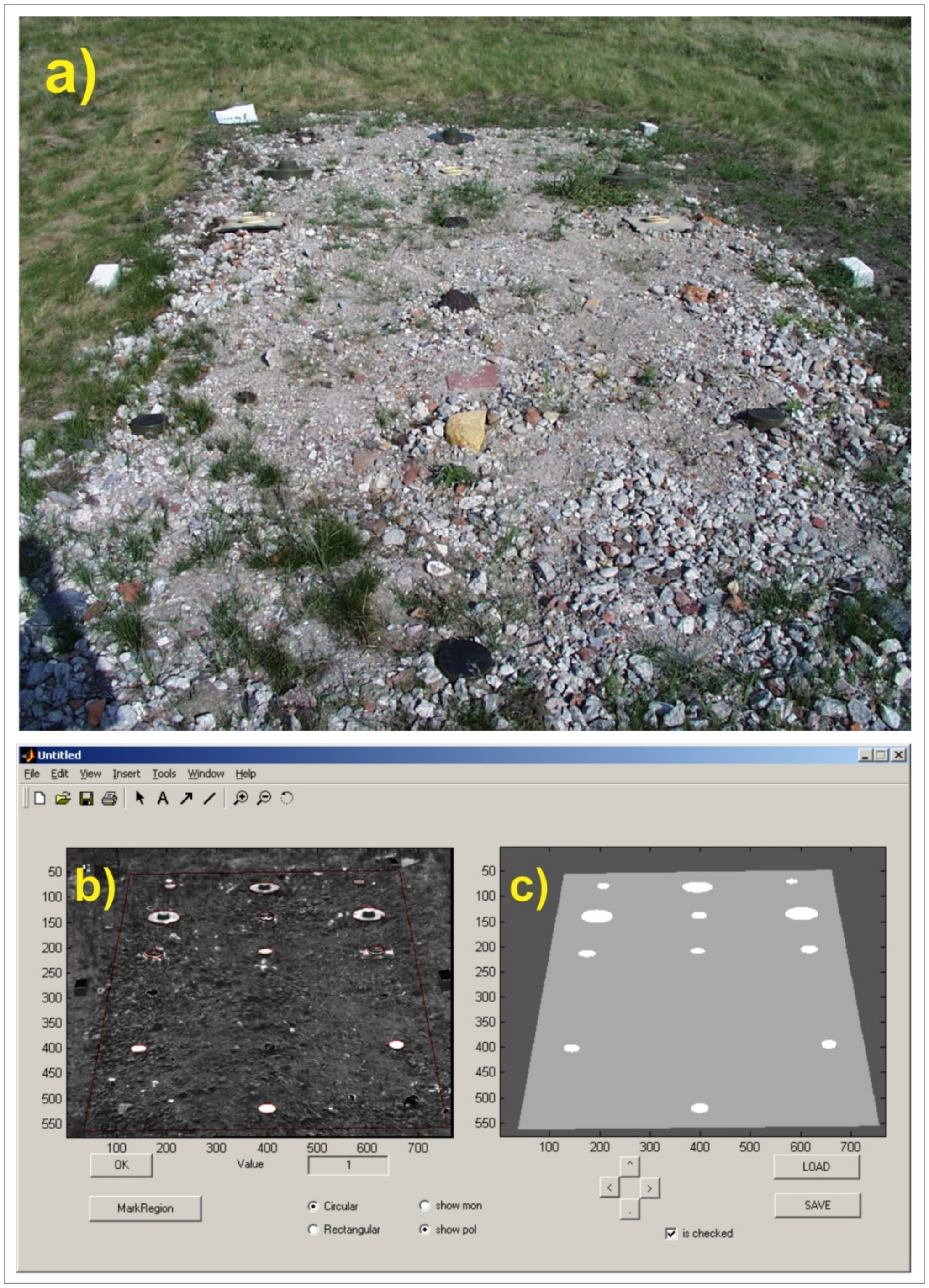


Figure 50 – The detection of land mines for humanitarian demining is one of the most important potential applications for DOLPi. I integrated this figure using images that were kindly supplied by TNO scientists Drs. Wim de Jong and John Schavemaker to exemplify their polarimetric imaging results of detecting replicas of four different types of mines that appear frequently in mine-afflicted countries. a) Visible image in a cluttered field. b) The mines can be clearly seen in the polarimetric image. c) Image showing the true location of land mines. Some of the clutter in the field reflects highly polarized light and may thus give some false positives to automatic software. Image credit: Images used with the author's permission from de Jong W, Schavemaker J, "Using Polarization Features of Visible Light for Automatic Landmine Detection" in Physics of Automatic Target Recognition - Advanced Sciences and Technologies for Security Applications, 3, 73-90, 2007.

The Need for Speed

The single largest shortcoming with DOLPi in its current incarnation is its slow frame rate. This is caused by the long time that it takes picamera to perform a single-shot frame capture. There are some tricks that I still have to try[15], but we'll probably have to wait for faster GPU firmware that will enable frame capture rates to produce smooth video even in preview mode.

The Raspberry Pi camera is able to acquire a set of frames quite quickly, so I tried using the camera's STROBE and FREX signals[16] to synchronize frame acquisition with changes in the VCPA's state. Unfortunately, activating these hardware signals needs to be done from the Raspberry Pi's GPU, which has closed-source firmware. I am not aware of anyone finding yet a hack to enable STROBE and FREX.

Low frame rates may not be a problem at all when imaging processes that change very slowly. DOLPi's frame rates more than suffice for capturing polarization images for medical diagnostics, atmospheric pollution monitoring, and scanning for hidden targets in a static scene. In fact, for many of these applications, even the frame rate of the mechanical filter-wheel-based DOLPi-MECH may suffice.

However, some promising applications of polarimetric imaging are contrast-enhancing techniques that enable the same type of real-time feats performed by animals with polarization-sensitive vision (e.g. navigation, camouflage breaking, object recognition, etc.). To make these useful, such as for underwater vision by autonomous submarines (AUVs), or to provide useful visual cues to the algorithms that drive autonomous ground vehicles, the polarimetric imaging system needs to be fast to allow real-time use by machine-vision algorithms.

Fortunately, Stoke's first two parameters (S_0=intensity, and $S_1=Q$=the difference in intensity accepted through polarizers at 0^o and 90^o to the horizontal) suffices for many of these applications such as distinguishing mud and water on roads, because they are based on the properties of light reflected by this type of surfaces (Figure 42) [Rankin and Matthies, 2008; Umansky, 2013]. Acquiring a single pair of orthogonal images is obviously faster than acquiring (and processing) three, so higher frame rates are possible with polarization-contrast imaging. A commercial example of such a device is Bossa Nova's Samba camera, which achieves up to 130 fps with a resolution of 320x240 pixels [Bossa Nova, 2010], and has been used by scientists at the Jet Propulsion Laboratory for daytime mud detection for Unmanned Ground Vehicle (UGV) autonomous navigation [Rankin and Matthies, 2008].

[15] For example, using a generator function within capture_sequence for changing the state of the VCPA as described in http://raspberrypi.stackexchange.com/questions/22040/take-images-in-a-short-time-using-the-raspberry-pi-camera-module.

[16] These lines are available in the Raspberry Pi camera clone made by Arducam (Rev.C OV5647).

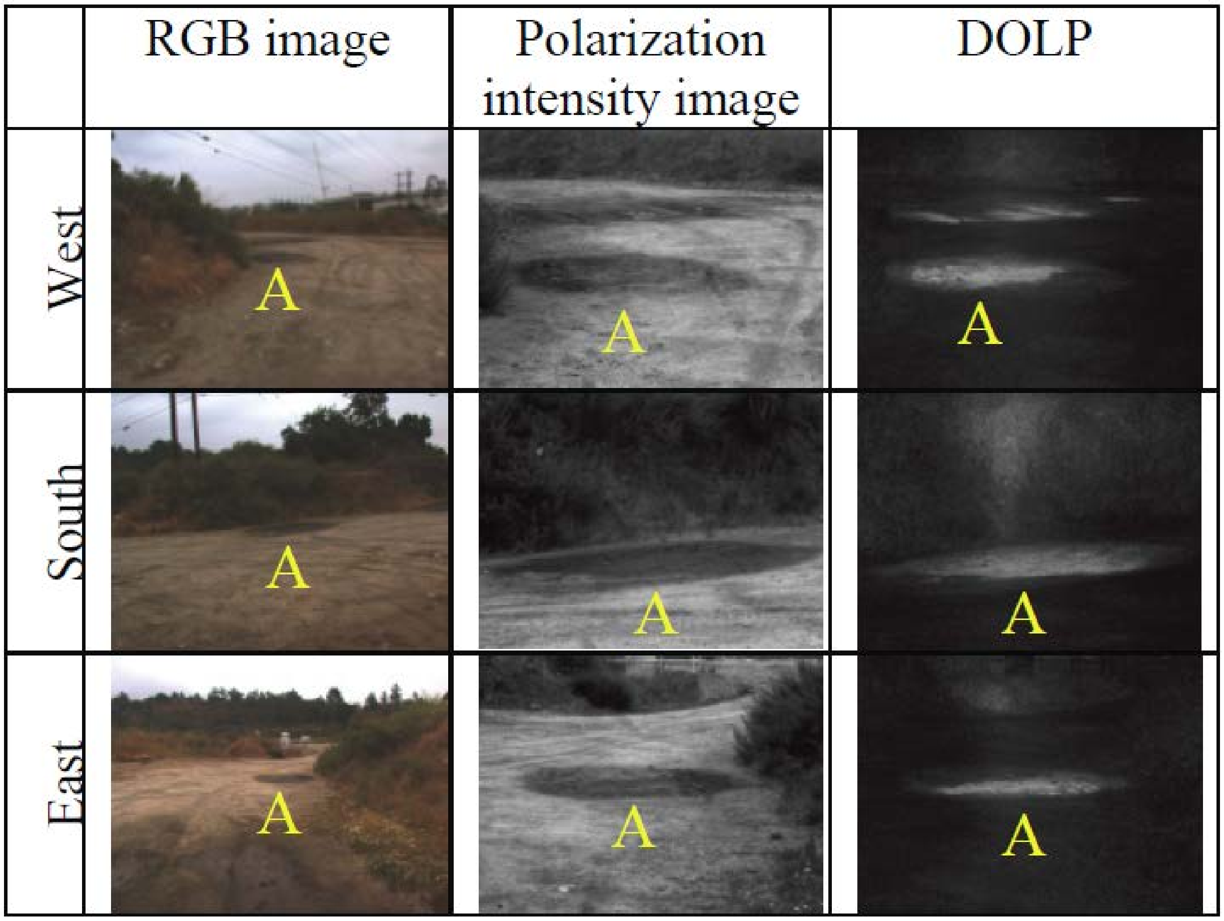


Figure 51 – Polarization imaging is an effective method of detecting water and mud on terrain, and is thus an attractive sensing modality to provide cues to the path-planning algorithms used in autonomous Unmanned Ground Vehicle (UGV). These images show a mud body (labeled "A") imaged on an overcast day from three directions with a color and a polarimetric camera. The DOLP on mud is consistently high, independent of sensor orientation, and is thus useful for detecting and localizing mud to be used by a UGV to plan safe paths. Image credit: work performed for the U.S. Government by Dr. Arturo Rankin of the JPL [Rankin and Matthies, 2008].

An easy way to demonstrate polarization difference imaging is to generate a color image by assigning the RGB channels as follows:

- Blue channel = intensity = (image0 + image90)/2
- Green = image0
- Red = image90

Non-polarized light will come through in grayscale, horizontally-polarized light will be colored green, while vertically-polarized light will be colored red. Table 8 is a Python listing for the code to implement a simple real-time contrast polarimeter with DOLPi. I achieve approximately 4Hz at a resolution of 320x240 pixels, the limiting factor again being the long time that it takes picamera to perform a single-shot frame capture. Figure 52 shows one frame of the real-time stream. Please remember that for my

DOLPi prototype (Figure 3), the analyzing polarizer film is placed at 45^o, so the horizontal frame of reference is tilted by 45^o.

Table 8 – DOLPi_DIF_RGB Python code. Produces a real-time image showing contrast of polarization

```
#   DOLPi_DIF_RGB.py
#
#   This Python program demonstrates the DOLPi polarimetric camera
#   running as a real-time contrast polarimeter. Images are displayed
#   in RGB format
#
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                               (MIT, opensource.org/licenses/MIT)
#
#   This version uses a voltage-controlled polarization analyzer (VCPA) that
#   consists of a liquid crystal panel(LCP) and a polarizer film.  The polarizer
#   film is placed between the LCP and camera.
#   The VCPA is driven by a GPIO pin-controlled AC driver to 2 states:
#   1. GPIO pin 22 high for no rotation of polarization
#   2. GPIO pin 22 low for 90 degrees of rotation
#
#import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import RPi.GPIO as GPIO
import numpy as np

def disp(image0,image90):
    # display function displays polarization information as an RGB image
    # encoded as:
    #   Blue channel = intensity = (image0 + image90)/2
    #   Green = image0
    #   Red = image90
    # function returns keyboard input in variable k
    stokesI=np.uint8((np.int16(image0)+np.int16(image90))/2)
    cv2.imshow("DOLPi",cv2.merge([stokesI,image0,image90]))
    k = cv2.waitKey(2)  #Check keyboard for input
    return k

#Define IO PINS
#--------------
dc0degpin=22   #VCPA driver control
GPIO.setwarnings(False)  #Don't issue warning messages if channels are defined
GPIO.setmode(GPIO.BCM)   #Initialize GPIO port
GPIO.setup(dc0degpin,GPIO.OUT)
GPIO.output(dc0degpin,False)
```

```
#
#Raspberry Pi Camera Initialization
#---------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
# Set resolution.  Higher resolution slows down acquisition
#camera.resolution = (128, 72)
camera.resolution = (320, 240)
#camera.resolution = (640, 480)
#camera.resolution = (1280,720)
camera.framerate=80
rawCapture = PiRGBArray(camera)
camera.led=False
#
#Auto-Exposure Lock
#------------------
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain
#
#Initialize flags
#----------------
loop=True  #Initial state of loop flag
first=True #Flag to skip display during first loop
video=True #Use video port?  Video is faster, but image quality is significantly
           #lower than using still-image capture

#Main loop
#---------
while loop:
   #grab an image from the camera at 0 degrees
   GPIO.output(dc0degpin,True)     #Drive LC panel with maximum amplitude
   time.sleep(0.02)              #Wait for LC panel to settle
   rawCapture.truncate(0)         #Capture image
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image0=rawCapture.array[:,:,1]  #Convert to gray scale (select blue channel)
   if first == False:            #If this is not first loop, then update image
      disp(image0,image90)
   #
   #grab an image from the camera at 90 degrees
   GPIO.output(dc0degpin,False)    #Drive LC panel with maximum amplitude
   time.sleep(0.02)              #Wait for LC panel to settle
   rawCapture.truncate(0)         #Capture image
```

```
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image90=rawCapture.array[:,:,1] #Convert to gray scale (select blue channel)
    first=False                  #First loop over
    k=disp(image0,image90)       #Update image and check for keyboard input
    #
    if k == ord('x'):   #Wait for x key to exit
        loop=False
#
#stokesI=np.uint8((np.int16(image0)+np.int16(image90))/2)    #Save last image
#cv2.imwrite("DOLPi_DIF_RGB.jpg",cv2.merge([stokesI,image0,image90]))
cv2.destroyAllWindows() #Close OpenCV windows
quit
```

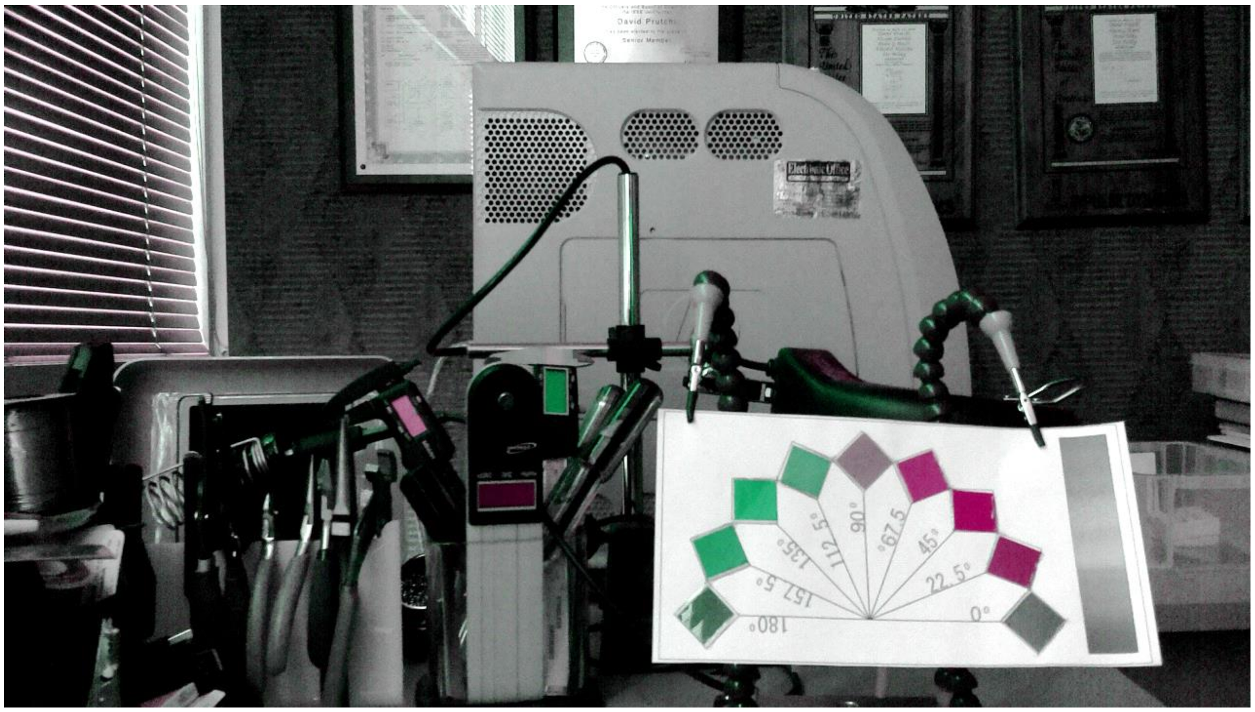


Figure 52 – Frame capture of the differential polarization imaging real-time stream produced by DOLPi running the code of Table 8. The analyzing polarizer film in my DOLPi prototype is placed at -45°, so the horizontal frame of reference is tilted by that amount. Non-polarized light is presented in grayscale, horizontally-polarized light (tilted by -45°) is be colored green, while vertically-polarized light (in the -45° of reference) is colored red.

For even higher polarimetric imaging speed, instead of trying to change the polarization state of an analyzer placed in front of a sensor, high-speed polarization imagers have been built using multiple sensors, each with its dedicated, fixed-state polarization analyzer. The simplest type consists of two imaging sensors placed behind orthogonal polarizing filters. As shown in Figure 54-a, this can be accomplished by placing two CCD cameras behind the output ports of a polarizing beamsplitter cube.

As shown in Figure 54-b, to acquire the complete linear Stokes parameter vector, a 3-way beamsplitting prism can be used to send copies of the scene to three CCD camera sensors, each placed behind a polarizer. This is the approach taken by FluxData for its FD-1665 polarization camera (http://www.fluxdata.com/imaging-polarimeters). I have been working on a DIY version of this camera by replacing the color filters of a 3-CCD color camera by polarization filters. My results so far are not very good, mostly due to the difficulty of taking apart the beamsplitter prism. I am currently building my own 3-way splitter and polarization filters to convert a 3-CCD JVC KY-F55BU color camera into an imaging polarimeter. In fact, I have been thinking about going retro all the way, and hacking a 3-tube (vidicon) color camera to make it easier to disassemble or replace the 3-way beamsplitter[17]. The advantage of starting from a 3-CCD (or 3-vidicon) camera is that the mechanical alignment hardware, as well as the image-combining circuitry is already built.

Here it is worthwhile mentioning a Matlab-driven polarimetric camera of this kind built by Mark Umansky [2013] for his M.Sc. thesis using multiple, synchronized machine-vision cameras viewing the scene through a DIY beamsplitter assembly (Figure 53). He built this camera to supplement the traditional sensor suite used in autonomous unmanned ground vehicles to identify patches of mud and water on a road to improve the navigation system's abilities.

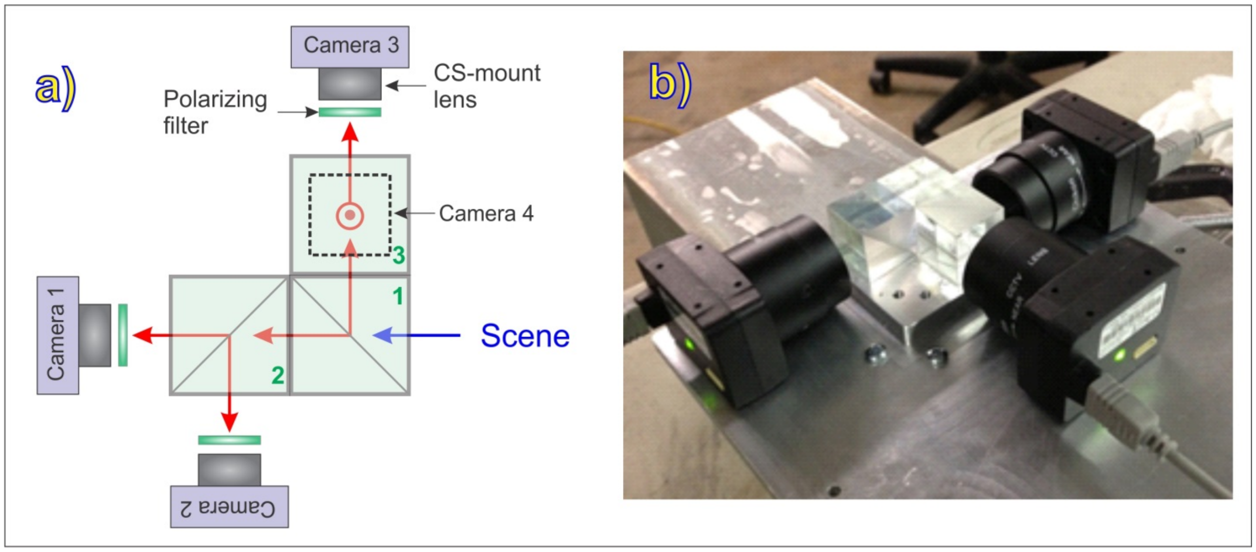


Figure 53 – In Mark Umansky's polarimetric camera, three beamsplitters are used to send the same image to up to four cameras placed behind polarizing filters (a). b) Umansky's 3-camera configuration showing the assembly of three ThorLabs 50/50 20mm non-polarizing beam splitters and three Point Grey Firefly MV cameras fitted with CS-mount lenses (a filter box holder placed over the beamsplitter completes the assembly). Adapted from Umansky M, *A Prototype Polarimetric Camera for Unmanned Ground Vehicles*, M.Sc. Thesis, Virginia Polytechnic Institute and State University, 2013. Used with permission from the author.

Lately, CCD imaging sensors have been fitted with micropolarizer arrays with linear polarizers (Figure 54-c) – much in the same way that color CCD image sensors are fitted with a color filter array. The pixelated polarization camera can then acquire multiple polarization orientations in a single video frame,

[17] http://blog.robindeits.com/2014/02/07/driving-a-viewfinder-from-1982/ describes the teardown of a vintage 3-tube JVC color camera. It looks like a promising model to modify into a polarimetric imager.

enabling instantaneous measurements of the linear Stokes parameters. This is probably beyond a DIY project, so we'll have to wait for one of the commercial outfits to mass-produce a low-cost CCD with integrated polarization filter array. Hopefully, DOLPi will muster enough attention for this to happen. If you really need this type of performance for your project right away, you could contact Moxtek (http://moxtek.com/optics-product/pixelated-polarizer/) or 4D Technologies (http://www.4dtechnology.com/products/polarcam.php).

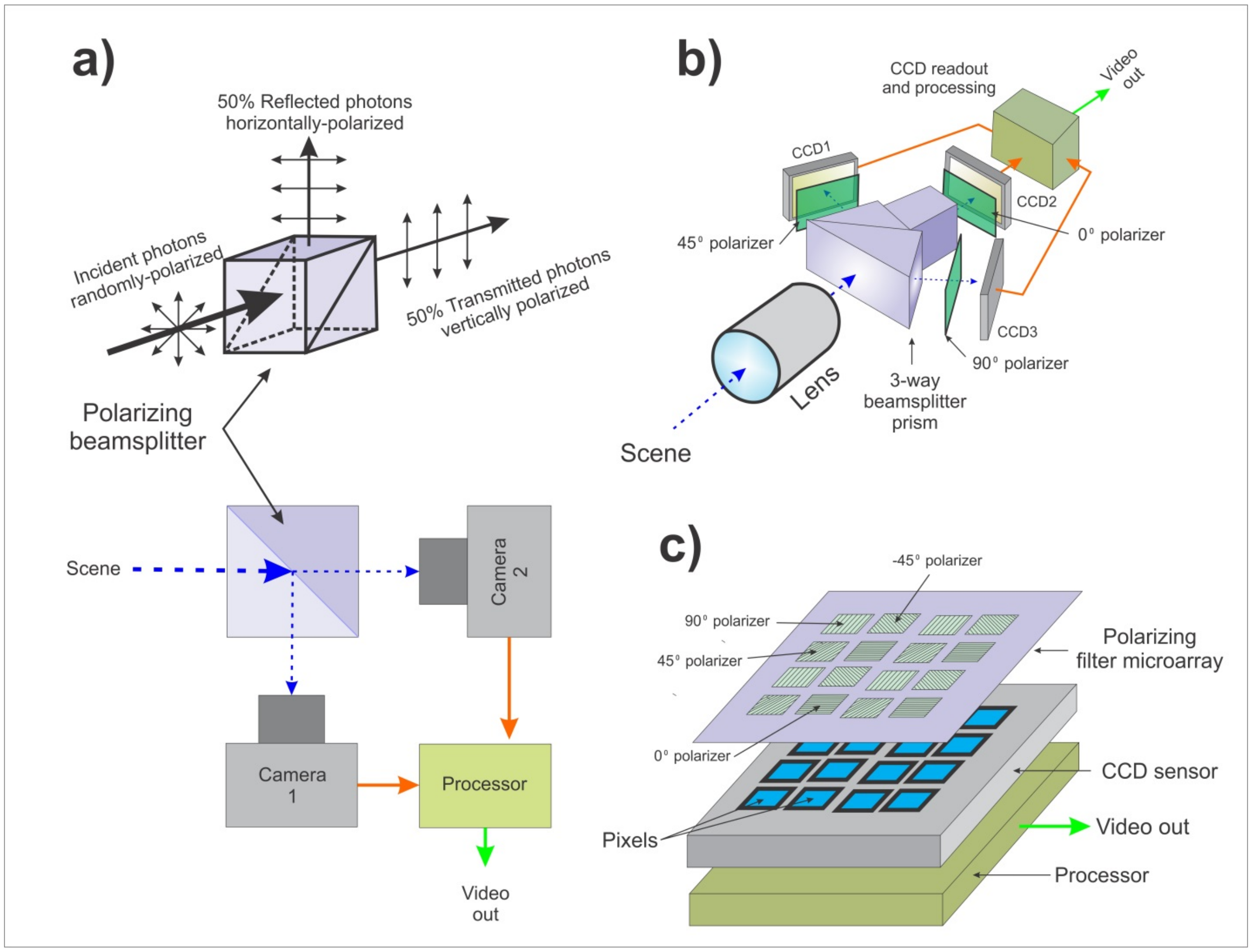


Figure 54 – Polarization imaging frame rate can be increased by using multiple sensors, each dedicated to a single angle of polarization analysis and then combining these into a single image through high-speed processing circuitry. a) In its simplest form, a measurement of two orthogonal polarizations can be used to calculate the first two Stokes parameters, which may suffice for many applications. b) The complete linear Stokes parameter vector can be obtained by routing copies of the scene to three separate image sensors via polarizing filters set at 0°, 45°, and 90°. c) Another possibility is to filter the individual pixels of an imaging sensor – much like it's done in a color camera, but using polarizing filters instead of the usual color filter array.

Lastly, Manakov et al. [2013] have proposed a very interesting idea for an add-on camera accessory that would go between the lens and camera of a standard DSLR to allow the camera to acquire High Dynamic Range (HDR), multispectral, polarization, or light-field pictures in a single shot. The idea is brilliant – take a kaleidoscope tube (similar to the toy, but with 4 mirrors instead of the usual 3) to copy the image so that 9 copies of the image fit the image sensor. Each of these copies is filtered differently, so the same sensor can produce all of the individual images needed for polarimetric (or multispectral, HDR, and light field) imaging in a single shot (albeit at lower resolution). Khlem et al. [2014] have applied for a patent on this "*KaleidoCamera*". The authors have posted a very informative video at https://youtu.be/NldRufqur3w. Until a commercial KaleidoCamera adapter becomes available, I intend to play with the technique using the Raspberry Pi camera as the sensor element.

Infrared Polarimetric Imaging

At wavelengths just beyond the visible spectrum, in the band known as the near-infrared (NIR), light experiences less scattering than visible light, allowing it to penetrate more deeply into scattering media like fog and biological tissues, making it useful in many fields ranging from biomedical imaging to imaging through fog and smoke. Furthermore, materials reflect NIR light differently than visible light, so IR imaging is often employed to gain contrast and improve detectability. Adding polarization information to NIR imaging further improves contrast, just as when it is done in the visible range.

However, DOLPi cannot work in the near-infrared unless specialized infrared polarizers are used. Figure 55-a and -b show on the left a stack of two pieces of standard film and on the right two IR polarizers. In Figure 55-a the polarizers are aligned, while –b shows them crossed. I took these pictures with infrared illumination (850 nm LED source) with a Canon T1i camera modified for infrared imaging (with 850 nm long-pass filter). As you can see, regular polarizing film is completely transparent to NIR radiation and cross-alignment does not cause any attenuation of the IR light. On the other hand, the polarizers meant for IR use (Thorlabs LPNIR100-MP2 at $984 each!) work as expected, allowing the IR light to pass through when aligned, and blocking it when crossed.

Placing the liquid-crystal panel between the crossed polarizers has the desired effect in both visible (Figure 55–c and –d) and IR light (Figure 55-e and –f). With no bias, the LCP twists the polarization by 90^o, allowing light to go through the crossed polarizers in both visible and IR spectra. However, light is blocked when the LCP is fully biased, but the contrast ratio for IR between the on and off states is not as strong as for visible light.

Super-expensive, lab-grade polarizer filters are not required to modify DOLPi or DOLPi-MECH for IR use because there are lower cost polarizing films that work in the NIR. For example, Edmund Optics' part number 33-082 is a 25 mm × 25 mm piece of wire-grid polarizing film that sells for $65. I sourced mine from MeCan Imaging from Japan, which sells wire-grid polarizing film made by Asahi KASEI (www.asahi-kasei.co.jp/ake-mate/wgf/en/product/index.html) directly from their website (www.mecanusa.com). An 80 mm × 240 mm sheet is $172, and an 80 mm x 50 mm sample is $55.

These films are produced by nano-scale printing a tight grid of extremely thin metallic wires onto a plastic base film. They work very well throughout the 400 nm (violet) to 1600 nm (NIR) spectrum, with consistent polarization performance that has very low wavelength dependence.

The Raspberry Pi camera also needs to be replaced for use in the NIR. Fortunately, the Raspberry NoIR camera is perfect for the application and connects just like the regular RasPi camera. However, since the polarization of reflections from materials in the scene is wavelength-dependent, and the fact that the camera's sensitivity is not flat once entering the IR spectrum, it is a good idea to use a bandpass filter to select the wavelength at which the polarimetric measurements are taken. A piece of KG5 heat absorbing glass (which is a typical infrared cut-off filter, and which can be harvested from an old digital camera) restores the NoIR to visible operation. Cheap IR low-pass filters are widely available on eBay® for DSLR IR photography. Typical band-stop wavelengths for IR filters are 720 nm, 760 nm, 850 nm, and 950 nm.

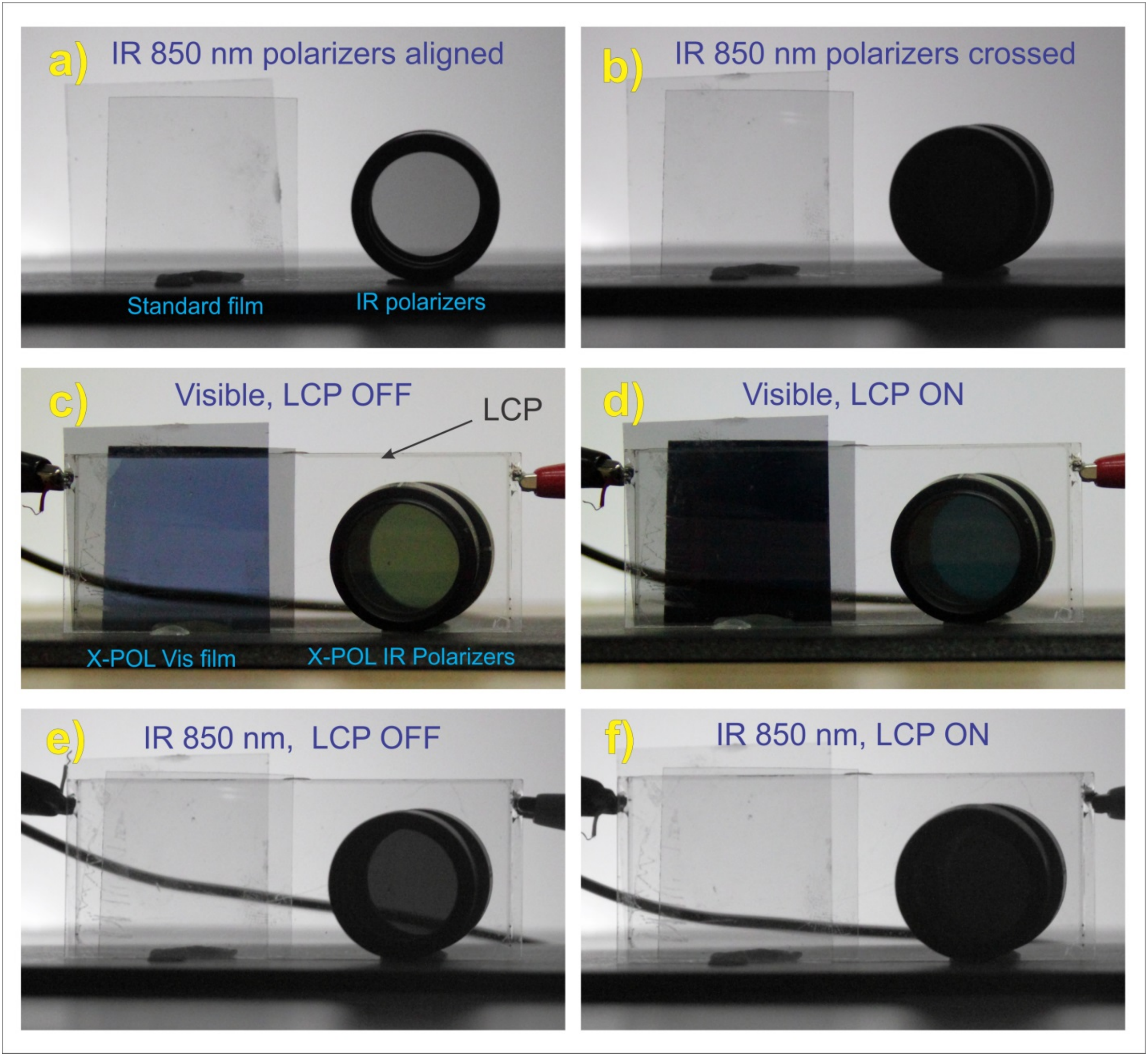


Figure 55 – Regular polarizer film doesn't work in the near-infrared, so relatively expensive, specialized polarizers are needed for NIR use: a) IR picture of aligned polarizers (regular polarizer film on left, dedicated IR polarizers on right). Placing the LCP used in DOLPi between cross-polarized filters does allow transmission in the visible and IR spectra when the LCP is held at 0V (c, e), and blocks light when fully biased (d, f), but the contrast ratio for IR between the on and off states is not as strong as for visible light. IR illumination source was a diy 10W dimmable IR LED flashlight [Prutchi, 2012-b].

At even longer wavelengths – like those in the long infrared spectrum - polarimetric imaging can really make a difference to improving detectability. For example, as shown in Figure 56, Polaris Sensor Technologies' eTherm™ vastly improves the contrast of thermal infrared images by using polarization to encompass information about the material itself, including its texture and shape [Tyo et al., 2006; Hanks et al., 2014].

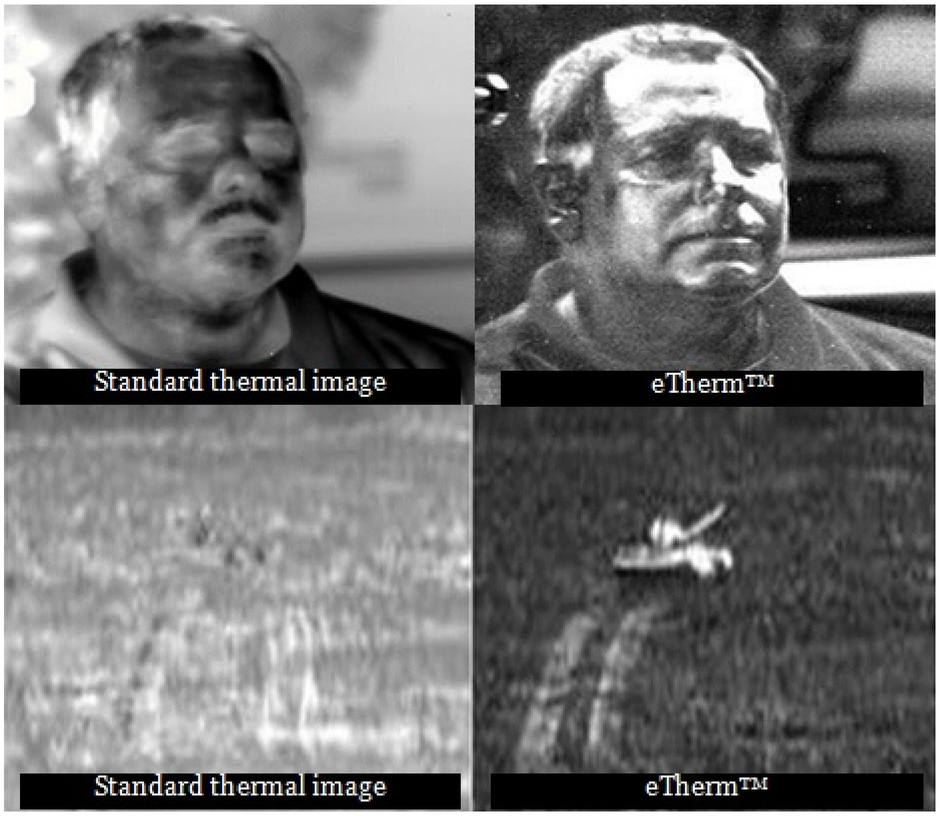


Figure 56 – Polaris Sensor Technologies' eTherm™ imaging vastly improves the contrast of thermal infrared images by using polarization to encompass information about the material itself, including its texture and shape. In the images on the bottom panels, a tank at thermal equilibrium with its surroundings provides very little contrast against its background for standard thermal imaging. On the other hand, the same scene image enhanced with polarization information dramatically improves the detectability of the target. Images courtesy of Polaris Sensor Technologies, Inc. For further information see PolarisSensor.com or contact (256) 562-0087 ext.2436. Images used with permission.

A diy longwave IR polarimetric imager could be built using Sparkfun's Lepton development kit and a wire polarizer. The FLIR Lepton chip is a complete LWIR camera that is ten times less expensive than a traditional thermal IR camera. It has a focal plane array (FPA) that consists of 80 × 60 active pixels sensitive to longwave IR radiation in the 8 to 14 µm wavelength range. The development kit (www.sparkfun.com/products/13233, $259.95) includes the Lepton chip and a breakout board that makes it easy to connect the camera to a Raspberry Pi or other platform via SPI. Unfortunately, wire-grid polarizers for these wavelengths are not cheap. The WP25M-IRC 1" diameter polarizer from Thorlabs costs around $1,300, and their WP25H-K - KRS-5 Holographic Wire Grid Polarizer retails for $1,400. Moxtek also produces wire grid polarizers specifically for thermal infrared applications [George et al., 2013], but again, these are not cheap.

Now, wire-grid polarizers often work as polarizing beamsplitters (Figure 54-a), so probably using a single wire-grid polarizer [Kudenov et al., 2008] and two Lepton modules would yield better results than rotating the polarizer. As shown in Figure 57, a diy thermal infrared polarimetric imager could be built using two FLIR Lepton modules to capture the vertically- and horizontally-polarized images split by the nanowire-grid polarizer[18]. These modules would need to be mechanically registered to present the same images to the Raspberry Pi (after mirror-imaging of one of the sensors in software). A low-cost Peltier-effect thermoelectric cooler would probably need to be used to prevent the cameras from recording IR radiation via the beamsplitter's secondary path. The Raspberry Pi would then process the images to calculate Stoke's first two parameters (S_0=intensity, and S_1=Q=the difference in intensity accepted through polarizers at 0^o and 90^o to the horizontal) as discussed above to output a polarization-contrast image that would provide enhanced detail over the standard thermal image.

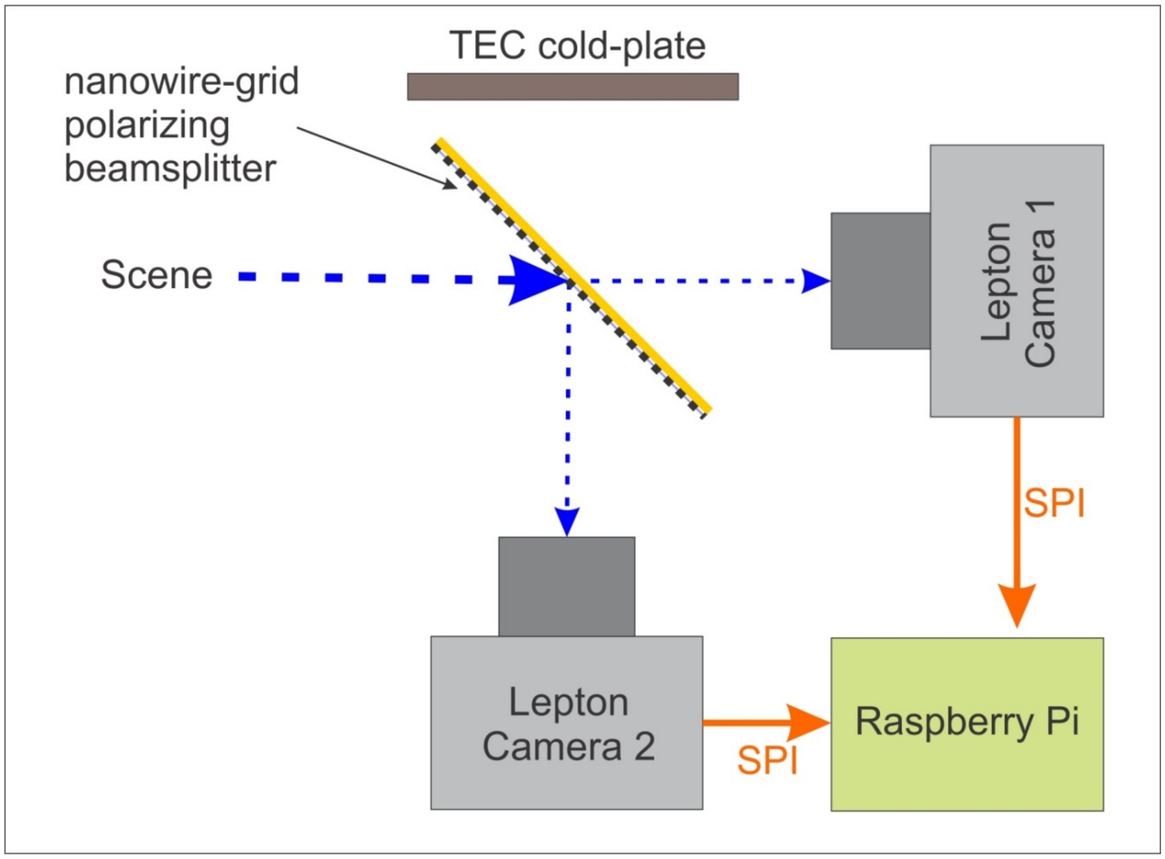


Figure 57 – Possible implementation of a diy long-wave (thermal) infrared polarimetric imager. Two FLIR Lepton modules are used to capture the vertically- and horizontally-polarized images split by the nanowire-grid polarizer. These are mechanically registered to present the same images to the Raspberry Pi (after mirror-imaging of one of the sensors in software). A Peltier-effect thermoelectric cooler is used to prevent the cameras from recording IR radiation via the beamsplitter's secondary path.

[18] However, one has to be careful with the way in which each of the images is treated by software because the image transmitted by a nanowire-grid beamsplitter has nearly 100% extinction, but this is not the case for the reflected image. Additionally, the design must take into consideration the so-called "Narcissus Effect", whereby the detector "sees" itself when reflected by the polarizing element.

Productizing DOLPi

The most effective way of productizing DOLPi would probably be to turn it into a Raspberry Pi hat like the one shown in Figure 58. The hat would be a printed circuit board that would both serve as a structural frame for the camera's subassemblies, as well as the substrate on which circuitry would be built. The DOLPi PCB would be sandwiched between the Raspberry Pi 2 and a 3.5 PiTFT touchscreen, much like in the prototype of Figure 26.

The DOLPi camera would be powered by an on-board flat Li-ion rechargeable battery placed onto the DOLPi hat PCB below the display. The recharge, LC panel driver, and autocalibration circuitry would be surface-mounted directly onto the hat. A custom-made 1.5" x 0.5" LC panel would be mounted within a dark plastic shield which would also incorporate space for a surface-mount LED and 45^o polarizer film for the autocalibration mechanism.

I estimate that the DOLPi hat could cost around \$25-\$30 to the manufacturer (in medium volumes, fully allocated), and could be sold to the public for around \$50. The buyer would then need to complete the assembly of the DOLPi by adding a Raspberry Pi 2 (\$40), SD card with pre-installed OS (\$10), Raspberry Pi camera board (\$30), and optionally a PiTFT display (\$35). A complete DOLPi would cost the user around \$165 and take about an hour to assemble with just a screwdriver. A more refined productized version of DOLPi is shown in Figure 59. It uses Raspberry Pi Foundation's new touchscreen display and a modified commercially-available enclosure. Complete construction details for this camera are presented in Appendix V.

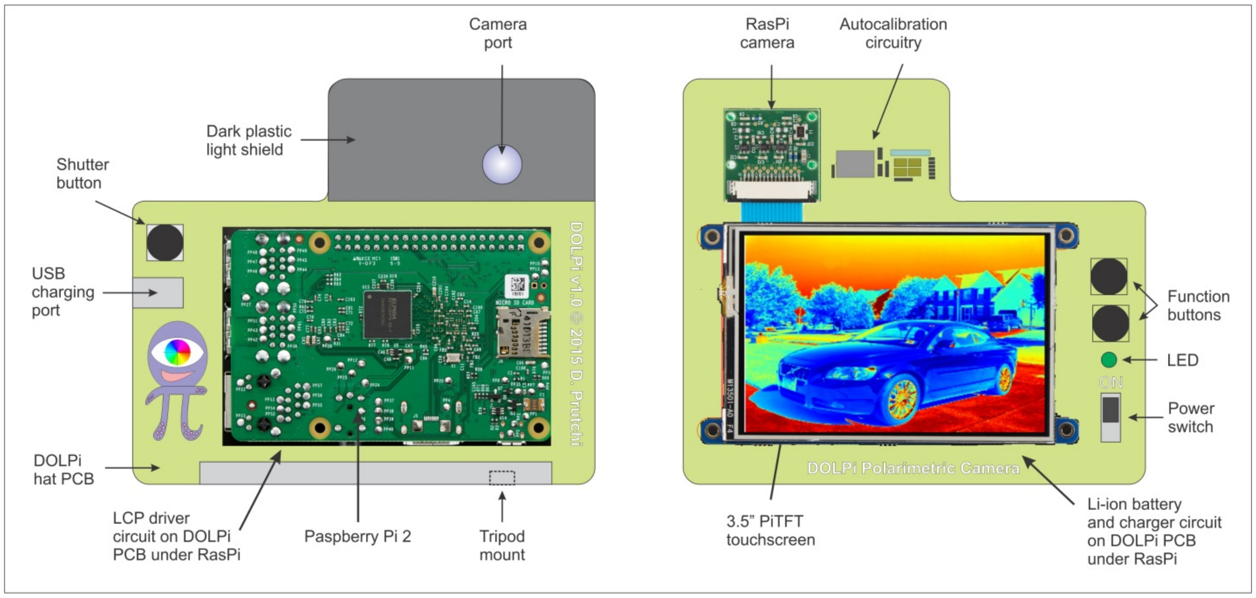

Camera port
Dark plastic light shield
Shutter button
USB charging port
DOLPi hat PCB
LCP driver circuit on DOLPi PCB under RasPi
Paspberry Pi 2
Tripod mount
DOLPi v1.0 © 2015 D. Prutchi
RasPi camera
Autocalibration circuitry
Function buttons
LED
ON
Power switch
DOLPi Polarimetric Camera
3.5" PiTFT touchscreen
Li-ion battery and charger circuit on DOLPi PCB under RasPi

Figure 58 – DOLPi can be productized as a low-cost Raspberry Pi hat. The DOLPi camera would be powered by an on-board Li-ion rechargeable battery. The recharge, LC panel driver, and autocalibration circuitry would be surface-mounted directly onto the hat. The Raspberry Pi 2 and 3.5" PiTFT touchscreen display would sandwich the DOLPi hat PCB. A custom-made 1.5" x 0.5" LC panel would be mounted within a dark plastic shield which would also incorporate space for a surface-mount LED and 45^o polarizer film for the autocalibration mechanism.

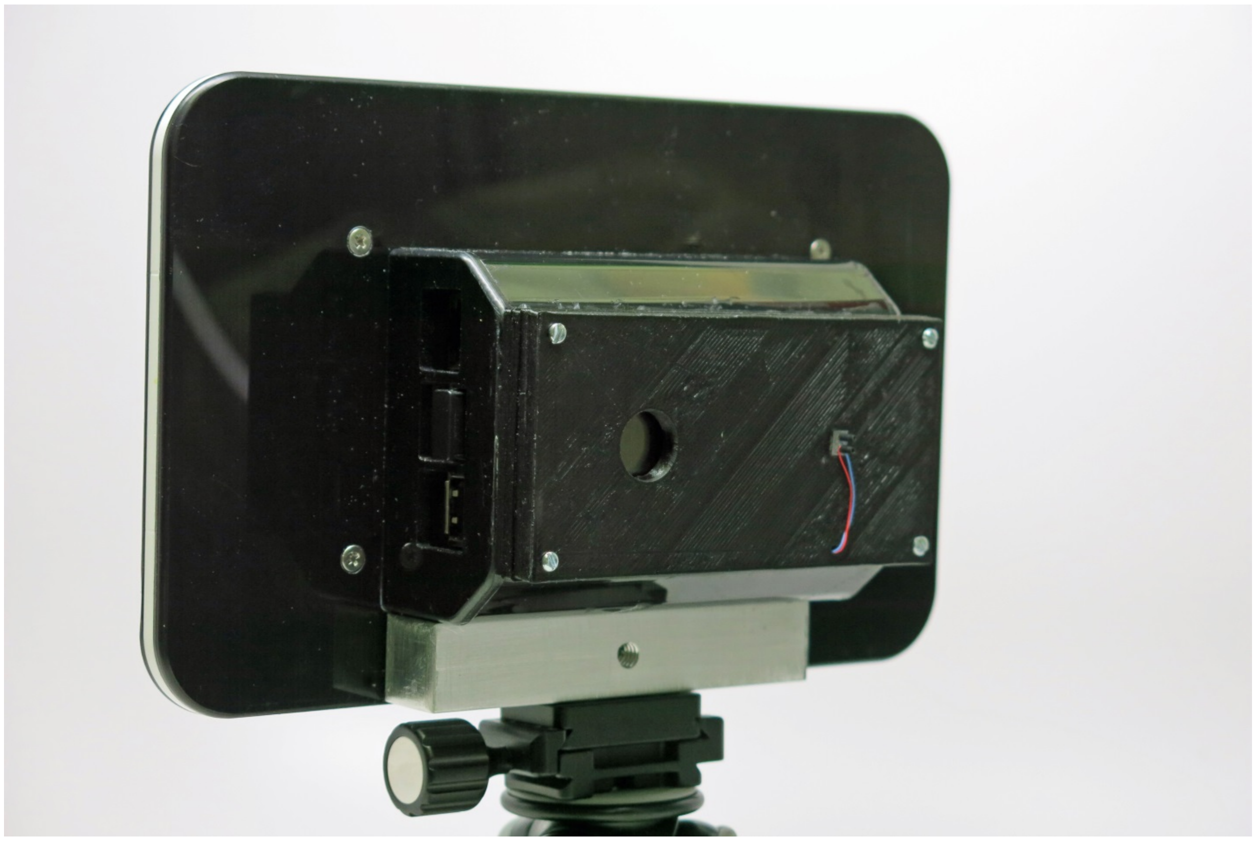


Figure 59 – Productized version of the electro-optic DOLPi polarimetric camera using Raspberry Pi Foundation's new touchscreen display and a modified commercially-available enclosure. Complete construction details for this camera are presented in Appendix V.

DOLPi-MECH can also be easily productized as a kit containing the parts necessary to build the camera. The kit would consist of:

- Laser-cut plastic base to hold the servo, Raspberry Pi, Raspberry Pi camera and tripod mount
- Aluminum tripod mount
- Laser-cut cardboard or plastic filter wheel
- Set of polarizer filters
- 180^o servo

The user would supply the Raspberry Pi 2, servo hat, camera and accessories.

In fact, the results I'm getting with DOLPi-MECH are so pleasing that I would actually productize DOLPI-MECH as an all-mode (multispectral + polarization) imager. Figure 60 shows my final productized prototype for this camera that I dub "DOLPi-UI" (UI stands for Universal Imager). Detailed construction instructions are provided in Appendix IV.

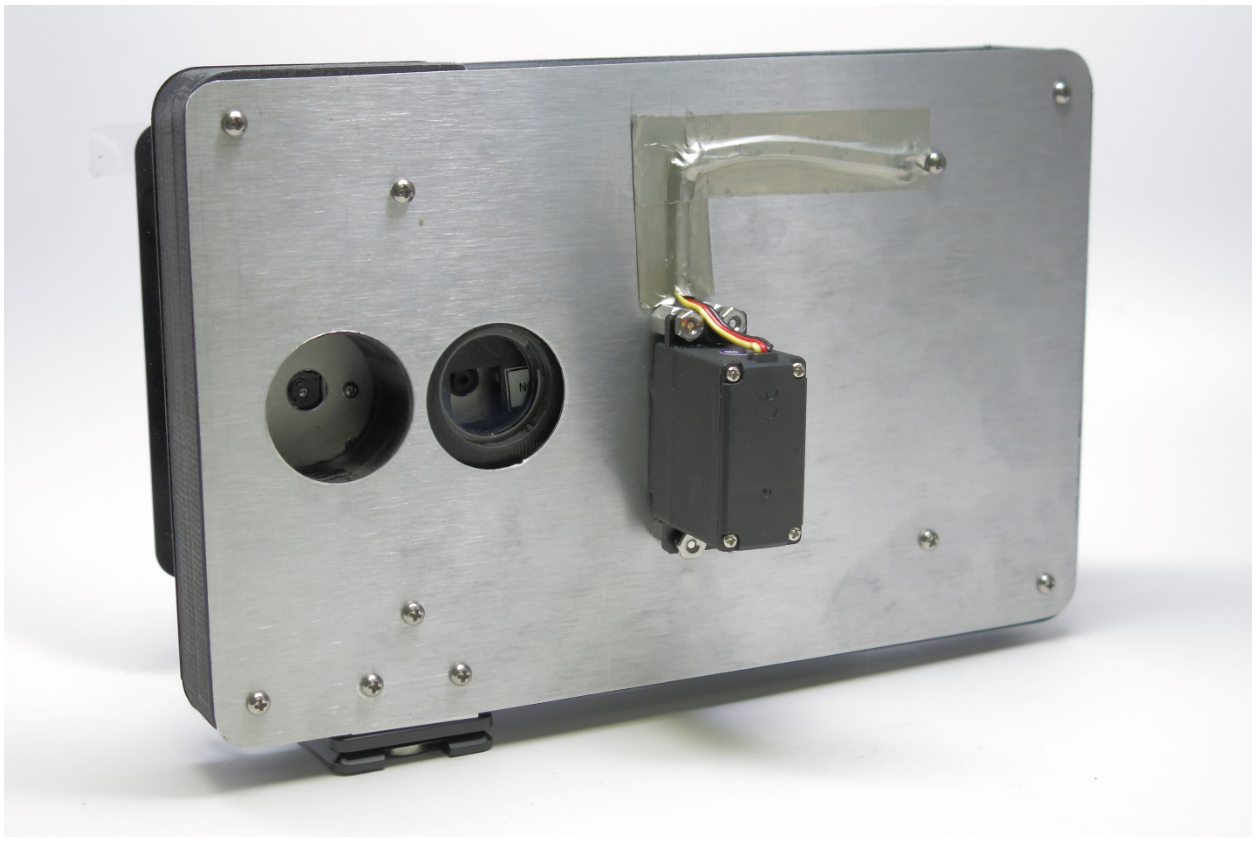


Figure 60 – Productized prototype of DOLPi-UI. This universal imager is capable of imaging and polarimetry in the near-infrared, visible, and near-ultraviolet regions of the spectrum. It also incorporates a FLIR Lepton® module to extend imaging (but not polarimetry) to the longwave infrared ("thermal" IR).

The idea would be to supply the components needed to build an all-mode imager (infrared + visible + ultraviolet + polarization) based on the RasPi NoIR camera, two servos, two filter wheels, a servo hat, and a RasPi 2 (Figure 61).

The polarization analyzer filter wheel would be rotated by a 90^o servo. It would be populated with the following filters:

1. Leave blank for non-polarized imaging
2. Wire-grid polarizer film at 0^o
3. Wire-grid polarizer film at 90^o
4. Wire-grid polarizer film at 45^o
5. Wire-grid polarizer film at -45^o
6. RH circular polarizer

The bandpass filter wheel would be populated with the following filters:

1. Leave blank for full-spectrum imaging
2. IR bandstop (e.g. Schott BG-39 glass) for visible-range imaging

3. IR longpass at 680nm
4. IR longpass at 780nm
5. Infrared-supressed near-UV (e.g. stack of Schott UG-1 + BG-39 glass filters)

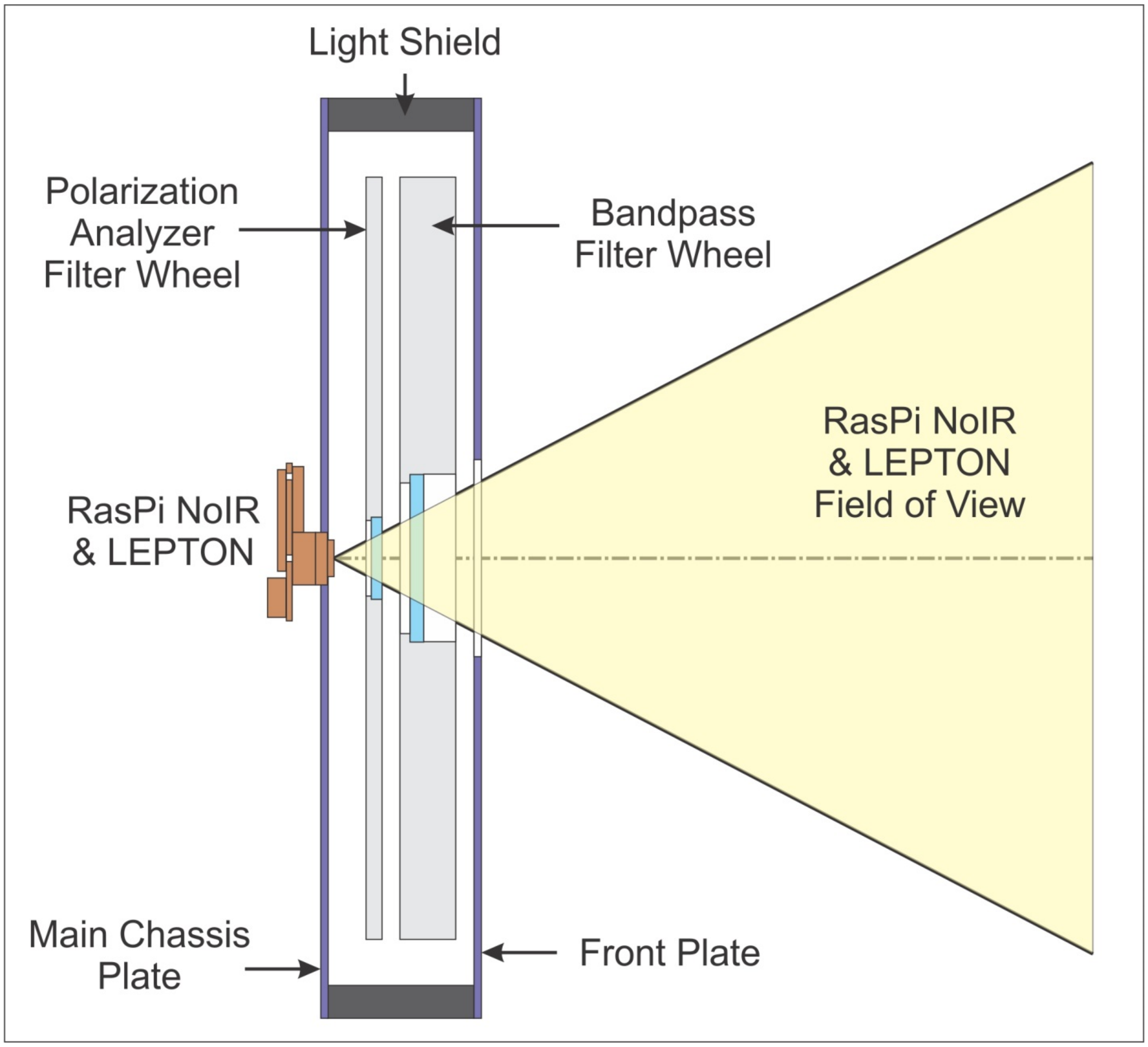


Figure 61 – Productized version of DOLPi-MECH would be an all-mode imager that allows the camera not only to analyze polarization, but also to operate as a multispectral imager (IR – VIS – UV). A second filter wheel is placed in front of the RasPi NoIR camera to select bandpass (visible, IR, or UV) for imaging, as well as for polarimetry. An optional FLIR Lepton® module can be used to supplement the imaging capabilities with longwave infrared ("thermal" IR),

Besides developing the above-mentioned hardware, productizing DOLPi would require developing user-friendly software to drive the cameras. My prototype software is straightforward and easy to understand for someone who has taken the time to read this paper. However, it is not consumer-ready. For that it would need to be developed to have an intuitive GUI, which would include touch-friendly features meant for a small screen.

Another possible product would be a simple polarization-switching visor for direct visualization of polarization states. As described previously, Boyd B. Bushman described in U.S. Patent 5,404,225 a pair of binoculars fitted with a VCPA to help detect man-made objects, such as military vehicles and high-voltage power lines. When viewed through the DOLPi-VISOR of Figure 62, man-made objects appear to flash as shown in Figure 19.

This device is very inexpensive and can be used as an alternative aid for locating landmines and unexploded munitions. Of course, it has many other applications where polarized reflections need to be highlighted above natural backgrounds. The performance of this simple visor is quite impressive since neither camouflage nor moderate foliage stops it from highlighting man-made targets because adequate flashing can still be observed.

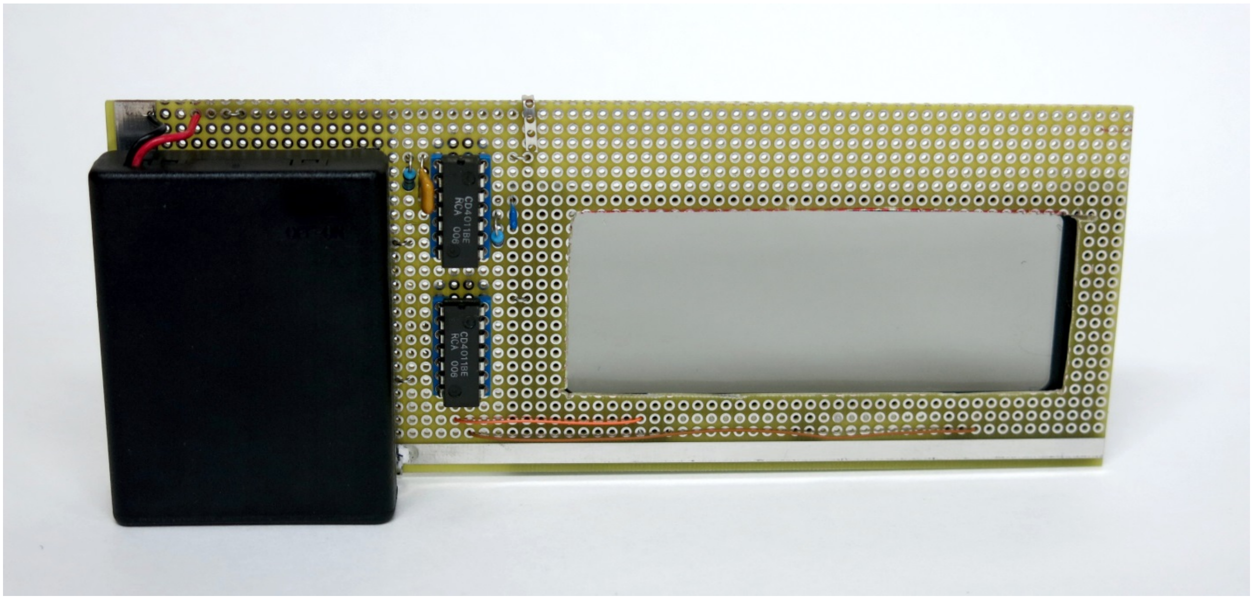


Figure 62 – DOLPi-VISOR is a direct polarization visualizer that makes polarized light blink. Man-made objects are thus clearly detectable against natural backgrounds even when hidden by camouflage and moderate foliage. Complete construction instructions for this simple visor are presented in Appendix VI.

Lastly, in developing DOLPi I considered relevant intellectual property (IP), and I believe that DOLPi, DOLPi-UI, and the DOLPI-VISOR can be productized and marketed without infringing on valid patents or other IP[19].

Saving the World

DOLPi will be there to save the day if Wonder Woman ever turns into a supervillain bent on destroying the world with her transparent jet[20]. Barring that scenario, DOLPi will probably serve more as a low-cost

[19] However, I am not a patent attorney and cannot render a binding FTO opinion on DOLPi. I expressly disclaim any liability for the infringement of any intellectual property by the making, using, or selling of devices based on the descriptions provided in this paper, and suggest that anyone interested in such projects seek proper legal counsel.

foundation for the development of applications based on polarization-sensitive imaging and machine vision. However, the disruptive power of this technology is truly awesome!

The use of polarimetric imaging for skin cancer detection was discussed above. Developing a low-cost self-examination camera to screen for skin cancers would have an enormous impact on world health. Just consider that according to the American Academy of Dermatology, one in five Americans will develop skin cancer in the course of a lifetime! A screening technique based on DOLPi would dramatically change the effect of this disease since basal cell and squamous cell carcinomas, the two most common forms of skin cancer, are highly curable if detected early and treated properly.

Polarimetric imaging has also started to be used for aircraft- and satellite-based characterization of atmospheric aerosol particles – a worldwide problem of significant impact. In fact, particle size, chemical composition, and shape can only be measured by including polarization information into a satellite's imaging capabilities [Snik et al., 2014]. Unfortunately, satellite observations only show what can be seen from above, and not what hides below layers of clouds and pollution.

DOLPi opens up the field of polarimetric imaging to the citizen scientist, enabling the development of techniques for remote sensing of vast swaths of atmosphere [Kreuter et al., 2013] to detect and characterize pollution and volcanic ash, giving the public unfiltered access to information about the nature and sources of hazardous air contaminants.

DOLPi also enables the development of low-cost tools for real-time monitoring of volcanic emissions, providing advanced data to populations at risk, as well as to local airports and air traffic such that appropriate warnings can be issued to those in harm's way. With a more strategic view, DOLPi gives researchers and enthusiast scientists the possibility of developing networks of low-cost sensors to measure atmospheric aerosol scattering and absorption properties that are presently the principal source of uncertainty in climate modeling.

Lastly, I would like to reiterate the enormous impact that DOLPi or DOLPi-MECH can make in humanitarian demining efforts. Landmines and unexploded ordinance pose a threat in nearly 80 countries around the world. Most at risk are children playing in the field and displaced civilians reconstructing their communities after armed conflict. Here, the broad availability and low cost of the components required to build a DOLPi camera would make it an ideal tool for use to rid the world of landmines and unexploded ordinance that are threatening civilian lives. Eliminating landmine casualties among returning refugees will give hope those who have endured the cruelty of war, and hence enhance the political and economic stability of those countries affected by landmines; and ultimately to promote worldwide peace and prosperity.

It is my hope that DOLPi will introduce fellow hackers and engineers to the awesome power of polarimetric imaging and inspire the development of breakthrough applications ranging from environmental monitoring and medical diagnostics, to security and antiterrorism applications.

[20] Presumably, the transparent jet would be transparent to color and intensity, but would modify the angle or phase of the electrical vector of the sky's polarized light, thus making it detectable by DOLPi.

References

Antonelli MR, Pierangelo A, Novikova T, DeMartino A, “Polarimetric Imaging of Human Tissue for Cancer Diagnostics: Experiment and Monte Carlo Modeling”, http://www.biop.dk/biophotonics09/Poster/Antonelli_NewSummary.pdf, 2009.

Bigué L, Ambs A, Jaulin A, Foulonneau A, Gendre L, Marconnet P, Imaging Liquid Crystal Polarimeters, 2008. Presentation slides available at: http://www.polarisation.eu/projectdir/Leiden_Bigue2.pdf

BirdLife International, “Migratory Soaring Birds Project Solar Energy Guidance V.1”, http://migratorysoaringbirds.undp.birdlife.org/sites/default/files/factsheet%20Solar%20Developer%20v1H.pdf, 2015.

Bossa Nova Technologies, *SAMBA Camera Brochure*, 2010. Available at: http://www.bossanovatech.com/Brochures/SAMBA%20camera%20-%20Brochure%202010.pdf

Bushman BB, *Object Detector*, U.S. Patent 5,404,225, Apr. 4, 1995. Available at: https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US5404225.pdf

DeHoog E, Luo H, Oka K, Dereniak E, Schwiegerling J, “Snapshot Polarimeter Fundus Camera”, *Applied Optics, 48(9)*, 1663–1667, 2009. Available at: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2853936/pdf/nihms183550.pdf

deJong W, Schavemaker J, “Using Polarization Features of Visible Light for Automatic Landmine Detection” in *Physics of Automatic Target Recognition - Advanced Sciences and Technologies for Security Applications*, 3, 73-90, 2007. Available at: http://publications.tno.nl/publication/102398/ckt8Xu/jong-2007-using.pdf

Driver PC, Fowler RE, *Method and Means for Enhancing Camouflaged Target Detection Utilizing Light Polarization Techniques*, U.S. Patent 3,352,965, Nov 14, 1967. Available at: http://www.google.com/patents/US3352965.

El-Saba A, Bezuayehu T, "Higher Probability of Detection of Subsurface Land Mines with a Single Sensor using Multiple Polarized and Unpolarized Image Fusion”, in *Polarization: Measurement, Analysis, and Remote Sensing VIII*, edited by Chenault, David B.; Goldstein, Dennis H., Proceedings of the SPIE, 6972, 2008.

George MC, Bergquist J, Wang B, Petrova R, Li H, Gardner E, "An Improved Wire Grid Polarizer for Thermal Infrared Applications", *Proc. SPIE 8613, Advanced Fabrication Technologies for Micro/Nano Optics and Photonics VI, Georg von Freymann; Winston V. Schoenfeld; Raymond C. Rumpf, eds.*, Vol. 8613, 86131I, 2013. Available at: http://moxtek.com/wp-content/uploads/pdfs/An-Improved-Wire-Grid-Polarizer-for-Thermal-Infrared-Applications.pdf

Hanks JB, Aycock TM, Chenault DB, “Investigating Clutter Reduction for Unmanned Systems Applications using Imaging Polarimetry”, *Proc. SPIE 9084, Unmanned Systems Technology XVI*, 90840U (June 3, 2014); doi:10.1117/12.2053078, 2014.

Horváth G, Blahó M, Egri A, Kriska G, Seres I, Robertson B, “Reducing the Maladaptive Attractiveness of Solar Panels to Polarotactic Insects”, *Conservation Biology*, 24(6), 1644-1653, 2010.

Horváth G, Barta A, Pomozi I, Suhai B, Hegedüs R, Åkesson S, Meyer-Rochow B, Wehner R, “On the Trail of Vikings with Polarized Skylight: Experimental Study of the Atmospheric Optical Prerequisites Allowing Polarimetric Navigation by Viking Seafarers”, *Phil. Trans. R. Soc. B*, 366, 772–782, 2011. Available at: http://rstb.royalsocietypublishing.org/content/royptb/366/1565/772.full.pdf

Jaulin A, Bigué L, “High Speed Partial Stokes Imaging using a Ferroelectric Liquid Crystal Modulator”, *Journal of the European Optical Society - Rapid publications*, 3, 2008 Available at: http://www.jeos.org/index.php/jeos_rp/article/view/08019

Klehm O, Ihrke I, Restrepo J, Manakov A, Hegeds R, PCT Patent Application WO2014124982 A1, *Plenoptic Imaging Device*, 2014. Available at:
https://www.google.com/patents/WO2014124982A1?cl=en&dq=WO2014124982&hl=en&sa=X&ved=0CBwQ6AEwAGoVChMI45yz1ov-xwIVRJoeCh0OKQtF

Kreuter A, Blumthaler M, “Feasibility of Polarized All-Sky Imaging for Aerosol Characterization”, *Atmos. Meas. Tech.*, 6, 1845–1854, 2013. Available at: http://www.atmos-meas-tech.net/6/1845/2013/amt-6-1845-2013.pdf

Kudenov MW, Dereniak EL, Pezzaniti L, Gerhart GR, “2-Cam LWIR imaging Stokes Polarimeter”, in *Polarization: Measurement, Analysis, and Remote Sensing VIII, edited by David B. Chenault, Dennis H. Goldstein, Proc. of SPIE*, Vol. 6972, 69720K, 2008. Available at:
http://fp.optics.arizona.edu/detlab/Articles-Publications/2008-Kudenov-2-Cam-LWIR-imaging-Stokes-polarimeter.pdf

Lefaudeux N, *Method and System for Stokes Polarization Imaging*, U.S. Patent 8,004,675, Aug. 23, 2011. Available at:
https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US8004675.pdf

Les C.B., “Polarization Vision Keeps Cuttlefish Safe”, *Photonics Showcase*, 7, July 2012.

Manakov A, Restrepo J, Klehm O, Hegeds R, Eisemann E, Seidel HP, Ihrke I, “A Reconfigurable Camera Add-On for High Dynamic Range, Multispectral, Polarization, and Light-Field Imaging”, ACM Trans. on Graphics (SIGGRAPH'13), 32(4), 2013. Available at: http://resources.mpi-inf.mpg.de/KaleidoCam/

Novikova T, Pierangelo A, Manhas S, Benali A, Validire P, et al., “The origins of Polarimetric Image Contrast between Healthy and Cancerous Human Colon Tissue”, Applied Physics Letters, American Institute of Physics (AIP), 102, 241103, 2013. Available at: https://hal.archives-ouvertes.fr/hal-00904992/PDF/Novikova_APL_2013.pdf

Prutchi D, Prutchi SR, *Exploring Quantum Physics through Hands-On Projects*, 335 pages, John Wiley & Sons, Inc., Hoboken, NJ, 2012.

Prutchi D, Prutchi AM, *d.i.y. High-Power, Swappable-Head, UV/IR/Visible LED Flashlight with Intensity Control*, http://www.prutchi.com/2012/04/27/d-i-y-high-power-swappable-head-uvirvisible-led-flashlight-with-intensity-control, 2012-b.

Rankin AL, Matthies LH, *Daytime Mud Detection for Unmanned Ground Vehicle Autonomous Navigation*, 2008. Available at:
https://www-robotics.jpl.nasa.gov/publications/Arturo_Rankin/asc2008-rankin-mud-detection-final.pdf

Schowengerdt RN, *Cloaking System Using Optoelectronically Controlled Camouflage*, U.S. Patent 5,307,162, Apr 26, 1994. Available at: http://www.google.com/patents/US5307162

Smith GS, "The Polarization of Skylight: An Example from Nature", *American Journal of Physics,* 75(1), 25-31, 2007. Available at: http://depa.fquim.unam.mx/amyd/archivero/AJP75_0025_1645.pdf

Snik F, Craven-Jones J, Escuti M, Fineschi S, Harrington D, de Martino S, Mawet D, Riedih J, Tyo JS, "An Overview of Polarimetric Sensing Techniques and Technology with Applications to Different Research Fields", *Conclusions from an Interactive, Multidisciplinary Workshop on Polarimetric Techniques and Technology at the Lorentz Center in Leiden*, the Netherlands, March 24-28, 2014. Available at: http://www.polarisation.eu/projectdir/Sniketal2014-polarimetryoverview-SPIE_9099-10.pdf

Tyo JS, Goldstein DL, Chenault DB, Shaw JA, "Review of Passive Imaging Polarimetry for Remote Sensing Applications", Applied Optics, 45(22), 2006. Available at:
http://www.coe.montana.edu/ee/jshaw/publications/PassiveImagingPolRS_Review_AO2006Aug.pdf

Umansky M, *A Prototype Polarimetric Camera for Unmanned Ground Vehicles*, M.Sc. Thesis, Virginia Polytechnic Institute and State University, 2013. Available at:
https://vtechworks.lib.vt.edu/bitstream/handle/10919/23724/Umansky_M_T_2013.pdf?sequence=1&isAllowed=y

Vedel M, Breugnota S, LechocinskiaN, "Full Stokes Polarization Camera", *SPIE Proc.* Vol. 8160-33, SPIE Optical Engineering + Applications, San Diego, Aug 21-25, 2011. Available at:
http://www.bossanovatech.com/Articles/Full%20Stokes%20Polarization%20Imaging%20camera%20SPIE%202011.pdf
and
http://www.polarisation.eu/projectdir/Full%20Stokes%20Polarization%20camera%20-%20Spatial%20inhomogeneity%20and%20field%20calibration%20-%20Leiden%202014.pdf

Wang Y, Kudenov M, Kashani A, Schwiegerling J, Escuti M, "Snapshot Retinal Imaging Mueller Matrix Polarimeter", Proc. SPIE 9613, Polarization Science and Remote Sensing VII, 96130A, 2015.

Appendix I – StokesParams.m - Matlab Function for Calculating and Displaying Stokes and Polarization Parameter Images

```
function StokesParameters = StokesParams (I90, I45, I0, option)
% Function StokesParams.m
% Calculates and displays Stokes and polarization parameter images
% ----------------------------------------------------------------
%
% Inputs:
%      Im90, Im45, Im0, Option
%      where:
%            Im0 = Image at 90 degrees
%            Im45 = Image at 45 degrees
%            Im0 = Image at 0 degrees
%      Option = 1 or 2
%      where:
%            Option = 1: Internally converts color RGB image to grayscale
%            Option = 2: Uses grayscale (UINT8) image arrays
%
% Calculates and displays Stokes parameter images
% -----------------------------------------------
% S0 = I
% S1 = Q
% S2 = U
%
% Calculates and displays polarization images
% -------------------------------------------
% Pol_int = Polarization Intensity
% Pol_deg = Degree of Linear Polarization (DoLP)
% Pol_angle = Angle of Polarization (AoP)
% Ih, Iv
if option ==1
    % Convert RGB images to grayscale double
    I90 = rgb2gray(im2double(I90));
    I0  = rgb2gray(im2double(I0));
    I45 = rgb2gray(im2double(I45));

elseif option ==2
    % Convert UINT8 grayscale images to double
    I90 = im2double(I90);
    I45 = im2double(I45);
    I0  = im2double(I0);
end

% Calculate Stokes parameters, I, Q, U (S0 S1 S2)
% I = I0 + I90
% Q = I0 - I90
% U = 2I45 - I0 - I90
StokesParameters.I = I0 + I90 ;
StokesParameters.Q = I0 - I90 ;
StokesParameters.U = 2.*I45  - I0 - I90 ;

% Polarization intensity = sqrt (Q^2 + U^2)
StokesParameters.Pol_int = sqrt(StokesParameters.Q.^2 +
StokesParameters.U.^2) ;
```

```
% Degree of Linear Polarization = Polarization intensity / I
StokesParameters.DoP = StokesParameters.Pol_int./StokesParameters.I ;
% Angle of Polarization = 0.5 * atan(U/Q)
StokesParameters.Pol_ang = 0.5.*
atan2(StokesParameters.U,StokesParameters.Q);
% Polarization ratio = Q/I
StokesParameters.Pol_ratio = StokesParameters.Q./StokesParameters.I ;
% Ih = 0.5(I+Q)
StokesParameters.Ih = 0.5.*(StokesParameters.I+StokesParameters.Q) ;
% Iv = 0.5(I-Q)
StokesParameters.Iv = 0.5.*(StokesParameters.I-StokesParameters.Q) ;

% Display source images and combine into RGB representation
figure;
subplot(2,2,1);
imshow(I90);
title('Red = Analyzer polarization: 90 degrees');
subplot (2,2,2);
imshow(I0);
title('Green = Analyzer polarization: 0 degrees')
subplot(2,2,3)
imshow(I45);
title('Blue = Analyzer polarization: 45 degrees')
subplot(2,2,4);
RGBpol(:,:,1)=I90;
RGBpol(:,:,2)=I0;
RGBpol(:,:,3)=I45;
imshow(RGBpol)
title('RGB Representation of Polarization')
axis('off')

%figure
%imshow(RGBpol)
%title('RGB Representation of Polarization')
%axis('off')

% Display Stokes parameter images
figure
subplot(2,2,1)
imagesc(StokesParameters.I,[0,1])
title('I (Stokes S0)');
colormap('jet')
colorbar
axis('off')
subplot(2,2,2)
imagesc(StokesParameters.Q,[0,1])
title('Q (Stokes S1)');
colormap('jet')
colorbar
axis('off')
subplot(2,2,3)
imshow(StokesParameters.U,[0,1])
title('U (Stokes S2)');
colormap('jet')
colorbar
axis('off')
```

```
% Display polarization parameter images
figure;
imagesc(StokesParameters.Pol_int)
colormap('jet')
title('Polarization Intensity')
colorbar
axis('off')

figure
imagesc(StokesParameters.DoP,[0,1])
colormap('gray')
title('Degree of Linear Polarization (DOLP)')
colorbar
axis('off')

figure
imagesc(StokesParameters.Pol_ratio)
colormap('gray')
title('Polarization Ratio');
axis('off')

figure
imagesc((StokesParameters.Pol_ang+(pi/2))*(180/pi),[0,180])
colormap('hsv')
title('Angle of Polarization (AoP)')
colorbar
axis('off')

figure;
subplot(1,2,1)
imagesc(StokesParameters.Ih)
colormap('jet')
colorbar
axis('off')
title('Ih')
subplot(1,2,2)
imagesc(StokesParameters.Iv);
colormap('jet')
colorbar
title('Iv')
axis('off')

% Display HSV components and representation
figure
HSV(:,:,1)=(StokesParameters.Pol_ang+(pi/2))/pi;
HSV(:,:,2)=StokesParameters.DoP;
HSV(:,:,3)=StokesParameters.Pol_int;
subplot(2,2,1)
imagesc(HSV(:,:,1),[0,1])
title('H = Angle of Polarization (AoP)')
colormap('jet'); axis('off')
subplot(2,2,2)
imagesc(HSV(:,:,2),[0,1])
colormap('jet'); axis('off')
title('S = Degree of Linear Polarization (DoLP)')
```

```
subplot(2,2,3)
imagesc(HSV(:,:,3),[0,1])
title('V = Polarization Intensity')
colormap('jet'); axis('off')
subplot(2,2,4)
imshow(hsv2rgb(HSV))
axis('off')
title('Polarization (HSV Representation)')

% Display HSV representation along colormap of HSV space
figure
subplot(1,2,1)
imshow(hsv2rgb(HSV))
axis('off')
title('Polarization (HSV Representation)')
subplot(1,2,2)
% Show color cone
imshow(imread('HSV_color_map.jpg'));
```

Appendix II - DOLPi Software Licenses

I'm making DOLPi software available under MIT license (MIT, opensource.org/licenses/MIT).

The following packages are used within DOLPI's Python code:

1. picamera - © Dave Jones. Free use and distribution permitted under conditions stated at: http://picamera.readthedocs.org/en/latest/license.html
2. cv2 - OpenCV is released under a BSD license and hence it's free for both academic and commercial use.
3. RPi.GPIO, time - Part of Raspbian. Free use and distribution, but a lawyer would be needed to decode the GPL and dozens of other licenses that cover all the packages of Debian (the Raspbian basis)
4. numpy - © The Scipy community. Free for use and distribution permitted under conditions stated at: http://docs.scipy.org/doc/numpy/license.html
5. Matplotlib - © John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the matplotlib development team, Matplotlib only uses BSD compatible code, and its license is based on the license: http://matplotlib.org/1.3.0/users/license.html
6. Adafruit_MCP4725 - © Adafruit Industries, BSD license: https://github.com/adafruit/Adafruit_MCP4725/blob/master/license.txt
7. Adafruit_ ADS1X15 - © Adafruit Industries, BSD license: https://github.com/adafruit/Adafruit_ADS1X15/blob/master/license.txt
8. I used an Adafruit 3.5" PiTFT display in my DOLPi prototype. This display has its own drivers and utility programs that are made available for free by Adafruit under the MIT and other licenses.

I prototype some of my software under MATLAB® which is a commercial package available for purchase from The Mathworks (http://www.mathworks.com/products/matlab/). The MATLAB® polarization imaging code shown in Appendix I of the of this paper is free for both academic and commercial use.

Appendix III – Polarization Test Target Pattern Template

This is the template for the polarization test pattern used to evaluate DOLPi. The pattern should be printed on 8 ½ x 11" vellum[21]. Polarizing film at the indicated orientations is adhered with clear double-sided tape. The rectangle polarizer filters are cut from linear polarizer film (Polarizing Film Sheet - set of 10 at $11.20 on Amazon.com), the two squares on the bottom are made of circular polarizer film taken from RealD 3D movie glasses (one square from each eye).

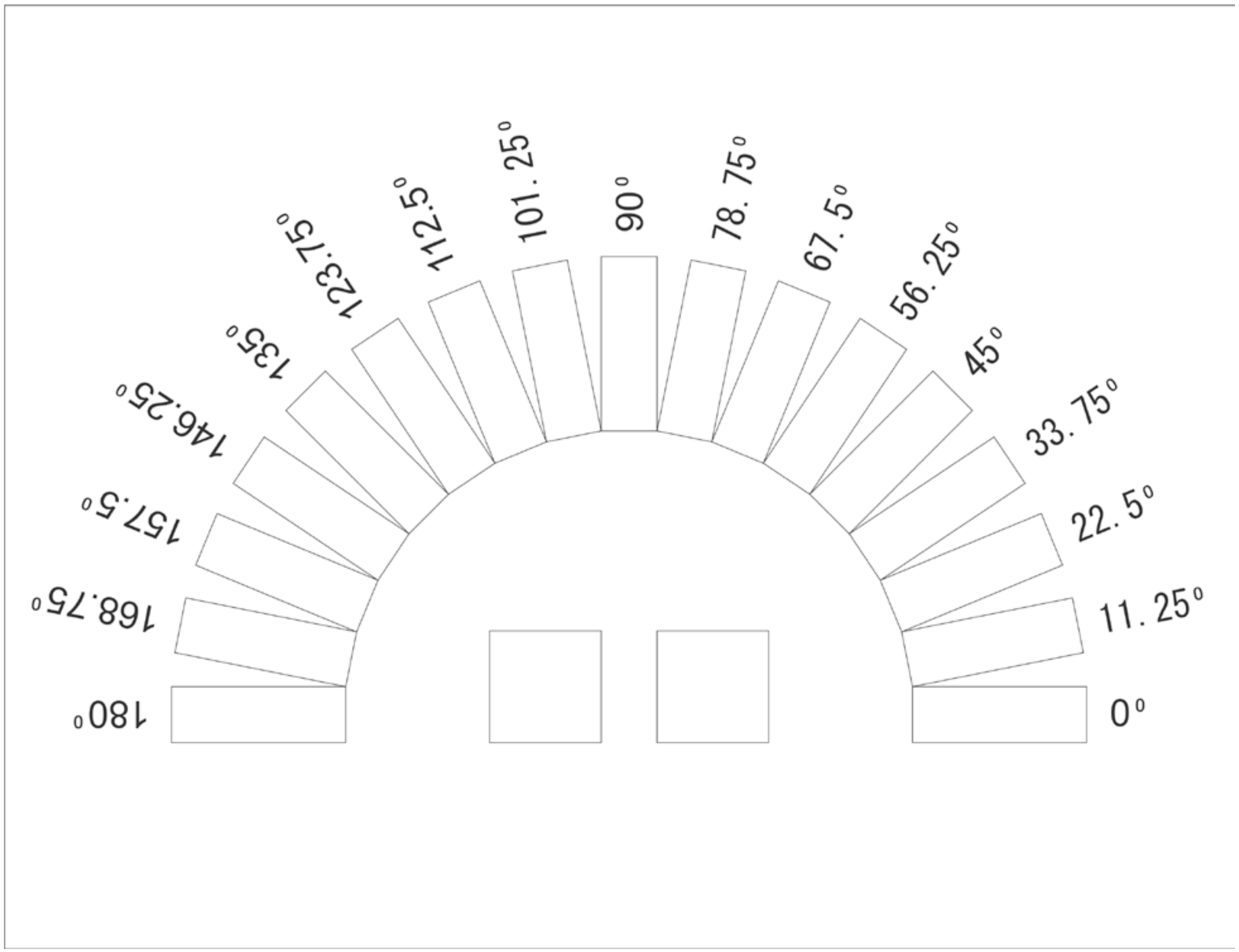


Figure 63 – Polarization test pattern template. This figure should be scaled to be printed on an 8 ½ x 11" piece of vellum.

[21] I use clear #17 vellum by Grafix (www.grafixarts.com)

Appendix IV – DOLPi-UI Construction

As described in the section entitled "Productizing DOLPi", I developed a final prototype suitable as the basis for manufacturing a filter-wheel-based all-mode camera capable of working in the infrared-visible-ultraviolet range both for imaging and polarimetric analysis. I added a FLIR Lepton® module to extend the camera's imaging capability (though not polarimetry) to the longwave IR ("thermal" IR).

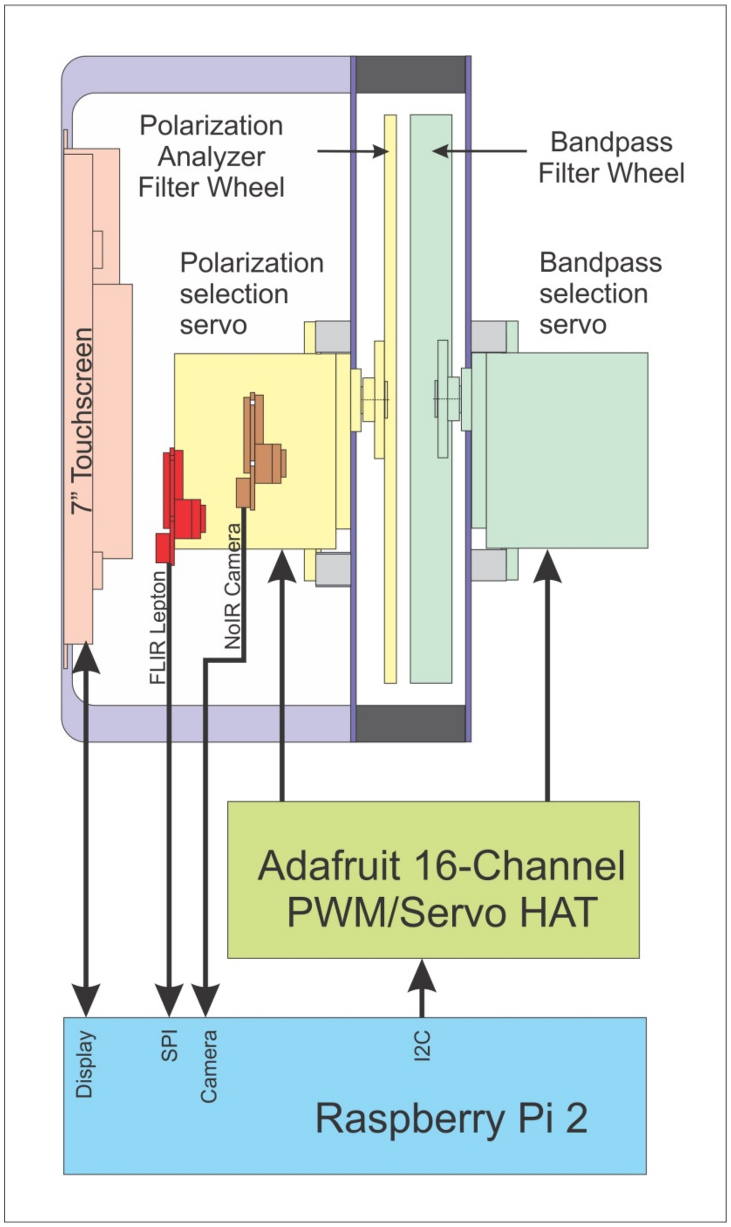


Figure 64 – Block diagram of the DOLPi-UI universal imager

Detailed construction steps allowing anyone to replicate this design are presented in this section. The parts list is shown in Table 9.

Table 9 - Bill of Materials for DOLPi-UI

Component reference	Component/Material	Source
RasPi	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	Adafruit ID:2358
SD Card	SanDisk SAEMSD64GBU3 64GB Extreme UHS-I microSDXC Memory Card (U3/Class 10)	B&H Photo
+5V Power Supply	5V 2A Switching Power Supply w/ 20AWG 6' MicroUSB Cable	Adafruit ID: 1995
RasPi NoIR Camera	Raspberry Pi NoIR Camera Board - Infrared-sensitive Camera	Adafruit ID: 1567
RasPi Touchscreen	Pi Foundation PiTFT - 7" Touchscreen Display for Raspberry Pi	Adafruit ID: 2718
Keyboard/Mouse	USB keyboard and mouse	Any suitable keyboard and mouse Preferred: Wireless Keyboard and Mouse Combo, Adafruit ID: 1738
USB cable	USB male to USB micro male 6" cable to connect Raspberry Pi to Pi Foundation PiTFT Touchscreen	Any suitable cable
Lepton® Longwave Infrared (LWIR) Imager	FLIR Lepton® Development Kit	Sparkfun Part No. KIT-13233
Wires to connect FLIR Lepton® break-out board to Servo Hat	Premium Female/Female Jumper Wires - 40 x 12" (300mm)	Adafruit ID: 793
EMI filter for Lepton®	Fair-Rite P/N 0443164251/B1	Newark Part No. 91F6476
Servo Controller	Adafruit 16-Channel PWM/Servo HAT for Raspberry Pi	Adafruit ID: 2327
2 x Servos	Two HiTec HS-322HD heavy-duty standard-size $\pm 90^{o}$ hobby servos	Jameco Part No. 395760
Optics		
4 x Wire Grid Linear Polarizer Film Filters	Four 14.5 mm x 14.5 mm filters cut from 80 mm x 50 mm Asahi KASEI wire-grid polarizing film sample.	MeCan Imaging
RHCP and LHCP Circular Polarizer Film Filters	RHCP and LHCP polarizer films taken from RealD 3D movie glasses	RealD movie glasses
VIS Filter	1" diameter BG-39 glass filter	Edmund Optics Catalog Number 48637
IR Filter	1" diameter optical-cast plastic IR filter	Edmund Optics Catalog Number 43948
UV Filter	1" diameter BG-39 glass filter + 1" diameter UG-1 glass filter	Edmund Optics Catalog Number 48637 + 46047
Filter Retention Rings	Easy-Install Spiral Internal Retaining Ring, Spring Steel, for 1" Diameter	McMaster-Carr Part Number 91663A570 (10 pack)
3D Printed Parts		
Polarization Analyzer	3D printed Polarization Analyzer Filter	DIY using FilterWheelSquare.stl

Filter Wheel	Wheel. Black PLA or ABS	
Bandpass Selection Filter Wheel	3D printed Polarization Analyzer Filter Wheel. Bandpass Selection Filter Wheel. Black PLA or ABS	DIY using FilterWheelCircle.stl
Light Shield Enclosure Section	3D printed Polarization Analyzer Filter Wheel. Light Shield Enclosure Section. Black PLA or ABS	DIY using LightShieldFrame.stl
Enclosure Back	3D printed Polarization Analyzer Filter Wheel. Enclosure Back. Black PLA or ABS	DIY using EnclosureBack.stl
Mechanical Hardware		
Front Plate and Main Chassis	Multipurpose 6061 Aluminum, Brushed Finish, 0.032" Thick. Front Plate and Main Chassis cut and drilled per templates	Machined from one sheet of McMaster-Carr Part Number 1651T11
Tripod Mount	Multipurpose 6061 Aluminum Rectangular Bar, 7/8" x 7/8" profile. Tripod mount machined per template.	Cut from McMaster-Carr Part Number 9008K13
4 x Chassis-to-Front Plate Standoffs	Four Aluminum Male-Female Threaded Hex Standoff, 1/4" Hex, 7/8" Length, 4-40 Screw Size	McMaster-Carr Part Number 93505A445
4 x Chassis-to-Touchscreen Standoffs	Four Aluminum Male-Female Threaded Hex Standoff, 1/4" Hex, 2" Length, 4-40 Screw Size	McMaster-Carr Part Number 93505A439
4 x Chassis-to-RasPi Standoffs	Four Aluminum Male-Female Threaded Hex Standoff, 1/4" Hex, 9/32" Length, 4-40 Screw Size	McMaster-Carr Part Number 93505A476
4 x RasPi to Servo Hat Standoffs	Four Aluminum Male-Female Threaded Hex Standoff, 1/4" Hex, 13/32" Length, 4-40 Screw Size	McMaster-Carr Part Number 93505A485
8 x Servo Standoffs	Eight Aluminum Male-Female Threaded Hex Standoff, 1/4" Hex, 11/32" Length, 4-40 Screw Size	McMaster-Carr Part Number 93505A482
¼" 4-40 screws	18-8 Stainless Steel Pan Head Phillips Machine Screw, 4-40 Thread, 1/4" Length	McMaster-Carr Part Number 91772A106 (100 pack)
RasPi Standoff Screws	18-8 Stainless Steel Pan Head Phillips Machine Screw, 4-40 Thread, 3/16" Length	McMaster-Carr Part Number 91772A105 (100 pack)
4/40 nuts	Type 18-8 Stainless Steel Hex Nut, 4-40 Thread Size, 1/4" Wide, 3/32" High	McMaster-Carr Part Number 91841A005 (100 pack)
LEPTON screws	18-8 Stainless Steel Flat Head Phillips Machine Screw, 1-72 Thread, 1" Length	McMaster-Carr Part Number 91771A175 (50 pack)
LEPTON nuts	Low-Strength Steel Hex Nut, Zinc Plated, 1-72 Thread Size, 5/32" Wide, 3/64" High	McMaster-Carr Part Number 90480A002 (100 pack)
RasPi NoIR camera	Nylon Pan Head Machine Screw,	McMaster-Carr Part Number

screws	Phillips, 2-56 Thread, 1" Length	93135A019 (100 pack)
RasPi NoIR camera nuts and LEPTON spacers	Nylon 6/6 Hex Nut, 2-56 Thread Size, 3/16" Wide, 1/16" High	McMaster-Carr Part Number 94812A100 (100 pack)
Hub screws and nuts	0-80 Thread Size machine screws	Ace Hardware
Miscellaneous		
Glue	White paper glue to retain polarizing film filters	Office supplies store
Double-sided tape	Heavy-duty double-sided tape to adhere EMI filter to Chassis plate	Office supplies store
Adhesive tape	½" adhesive tape to tack front servo wire to Front Plate	Office supplies store

Main Chassis Assembly

1. Cut a 220 mm x 140 mm plate of 0.032" thick aluminum.
2. Drill the Main Chassis Plate per the drill guide in "MainChassisPlateDrillGuide.pdf".
3. Round the corners per the drill guide in "MainChassisPlateDrillGuide.pdf".
4. Assemble the following components onto the Main Chassis Plate as shown in Figure 65:
    a. Raspberry Pi 2 with its SD Card and Adafruit 16-Channel PWM/Servo HAT for Raspberry Pi
    b. HiTec Servo
    c. RasPi NoIR camera
    d. FLIR Lepton® Development Kit
5. Wire modules to the RasPi.
6. Wire the FLIR Lepton® to the corresponding auxiliary pass-through pins on the Adafruit 16-Channel PWM/Servo hat. Follow the signal connections shown in: https://learn.sparkfun.com/tutorials/flir-lepton-hookup-guide
7. Assemble the touchscreen (without installing the RasPi over the screen driver board) as shown in http://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display

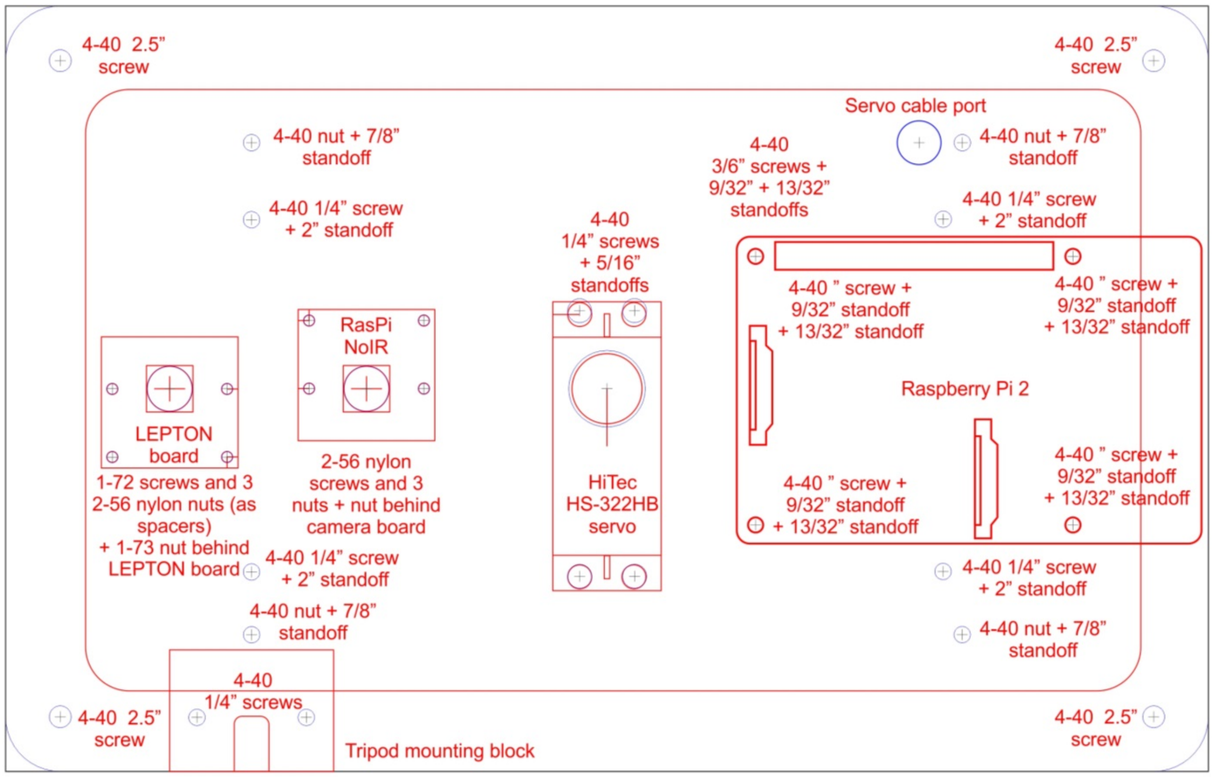

4-40 2.5" screw
4-40 2.5" screw
Servo cable port
4-40 nut + 7/8" standoff
4-40 3/6" screws + 9/32" + 13/32" standoffs
4-40 nut + 7/8" standoff
4-40 1/4" screw + 2" standoff
4-40 1/4" screws + 5/16" standoffs
4-40 1/4" screw + 2" standoff
4-40 " screw + 9/32" standoff + 13/32" standoff
4-40 " screw + 9/32" standoff + 13/32" standoff
RasPi NoIR
Raspberry Pi 2
LEPTON board
1-72 screws and 3 2-56 nylon nuts (as spacers) + 1-73 nut behind LEPTON board
2-56 nylon screws and 3 nuts + nut behind camera board
HiTec HS-322HB servo
4-40 " screw + 9/32" standoff + 13/32" standoff
4-40 " screw + 9/32" standoff + 13/32" standoff
4-40 1/4" screw + 2" standoff
4-40 1/4" screw + 2" standoff
4-40 nut + 7/8" standoff
4-40 nut + 7/8" standoff
4-40 1/4" screws
4-40 2.5" screw
Tripod mounting block
4-40 2.5" screw

Figure 65 – Assembly guide for the DOLPi-UI Main Chassis Plate

Front Plate Assembly

1. Cut a 220 mm x 140 mm plate of 0.032" thick aluminum.
2. Drill the Front Plate per the drill guide in “FrontPlateDrillGuide.pdf”.
3. Round the corners per the drill guide in “FrontPlateDrillGuide.pdf”.
4. Assemble a HiTec Servo as shown in the guide of Figure 66.
5. Route the servo wire through the servo wire hole in both the Front Plate and Main Chassis Plate.
6. Bend cable and tack with tape for neat appearance.

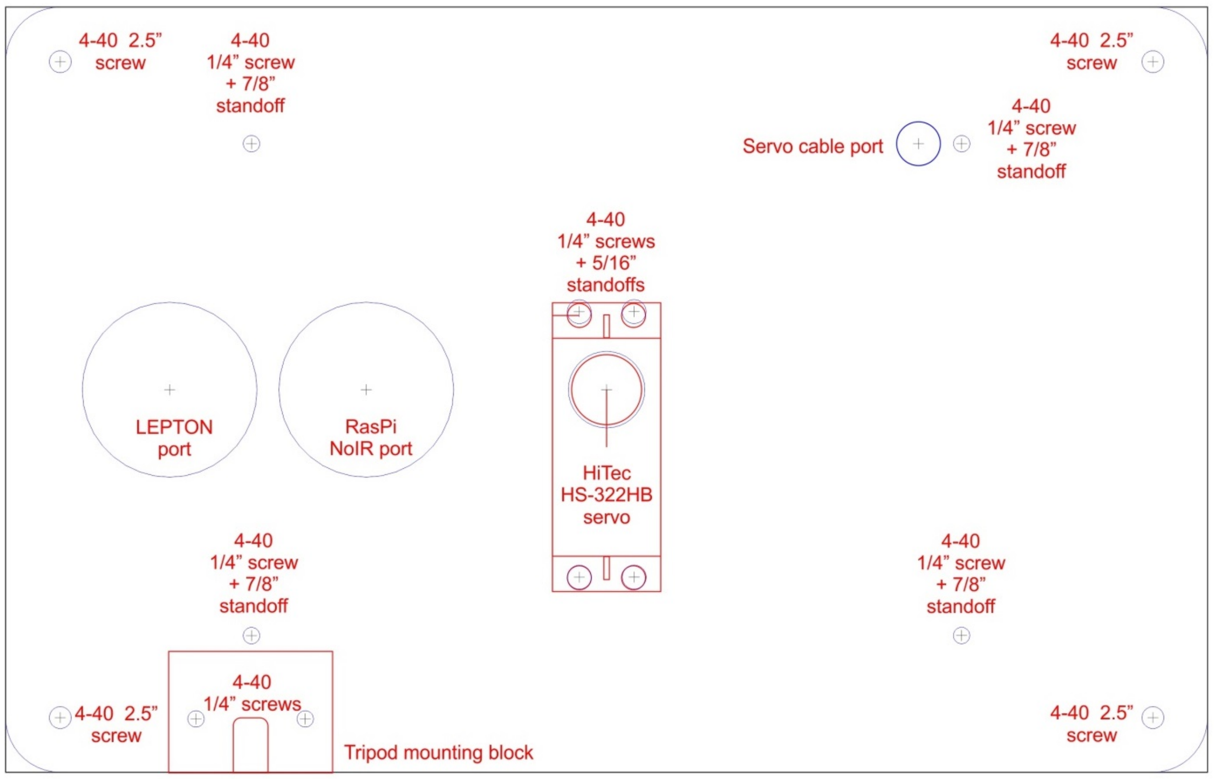


Figure 66 - Assembly guide for the DOLPi-UI Front Plate

Polarization Analyzer Filter Wheel Construction

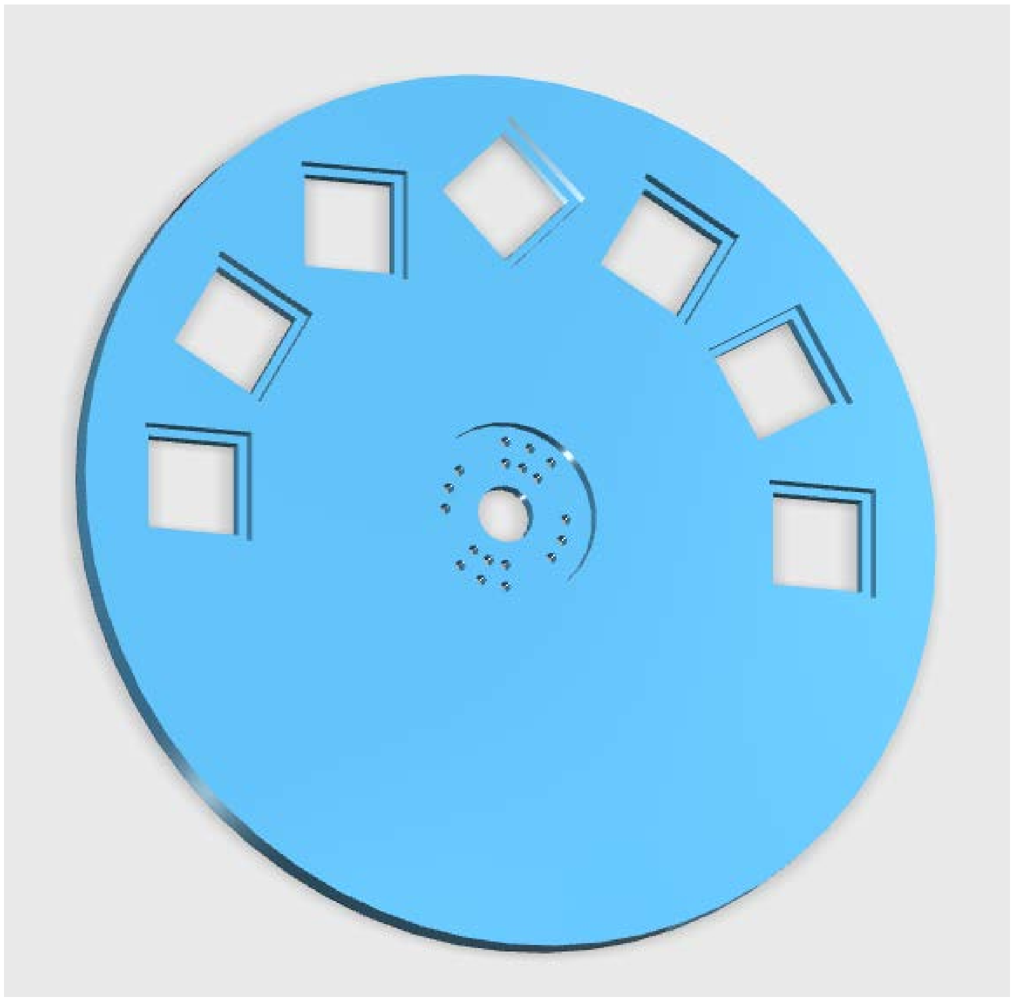


Figure 67 – Solid model rendering of FilterWheelSquare.stl

1. 3D-print FilterWheelSquare.stl using your printer's high-resolution mode. Figure 67 shows a rendering of the filter wheel.
2. Install the circular servo hub that came with the HiTec HS-322HD . Affix using two 0-80 machine screws and nuts.
3. Cut four 14.5mm x 14.5 mm squares of linear Polarizing Wire Grid Film. Install them onto the filter wheel as shown in Figure 68.
4. Cut a 14.5 mm x 14.5 mm square of the circular polarizer film from the right-eye lens of a pair of RealD 3D movie glasses. Install as shown, making sure that the linear-polarizer portion of the film will face the camera.
5. Cut a 14.5 mm x 14.5 mm square of the circular polarizer film from the left-eye lens of a pair of RealD 3D movie glasses. Install as shown, making sure that the linear-polarizer portion of the film will face the camera.
6. Hold the film squares in place using small drops of white paper glue on the filter corners.

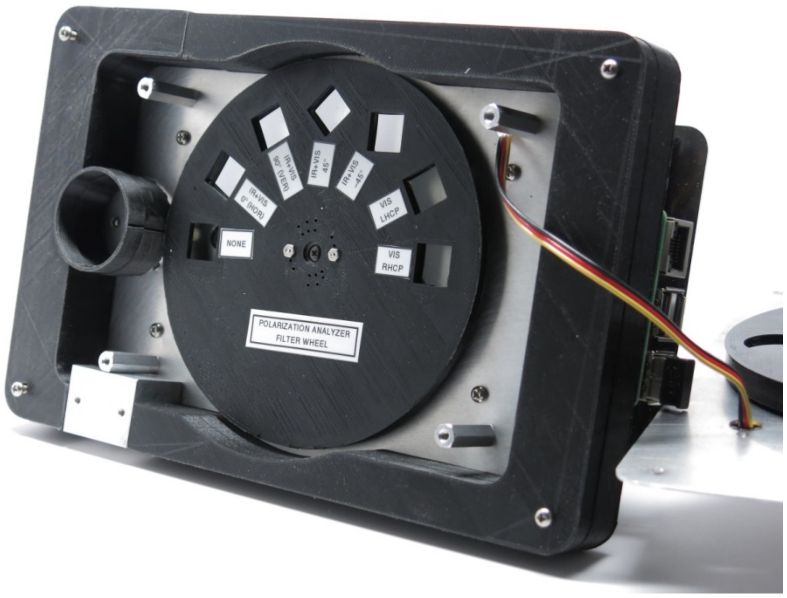


Figure 68 – Polarization filter wheel assembly showing orientation of polarizer films.

Bandpass Selection Filter Wheel Construction

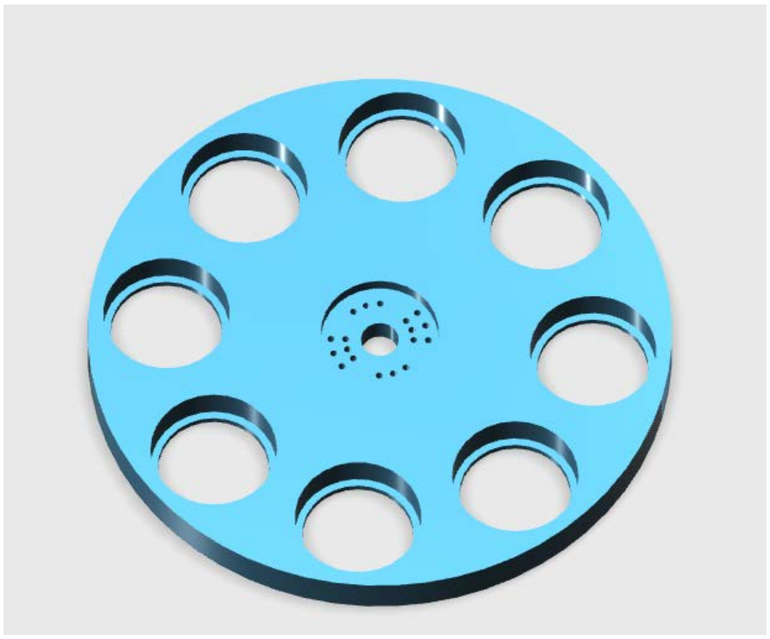


Figure 69 - Solid model rendering of FilterWheelCircle.stl

1. 3D-print FilterWheelSquare.stl using your printer's high-resolution mode. Figure 69 shows a rendering of the filter wheel.
2. Install the circular servo hub that came with the HiTec HS-322HD . Affix using two 0-80 machine screws and nuts.

3. Populate the filter wheel as shown in Figure 70.
4. Hold the filters in place using easy-install steel retention rings.

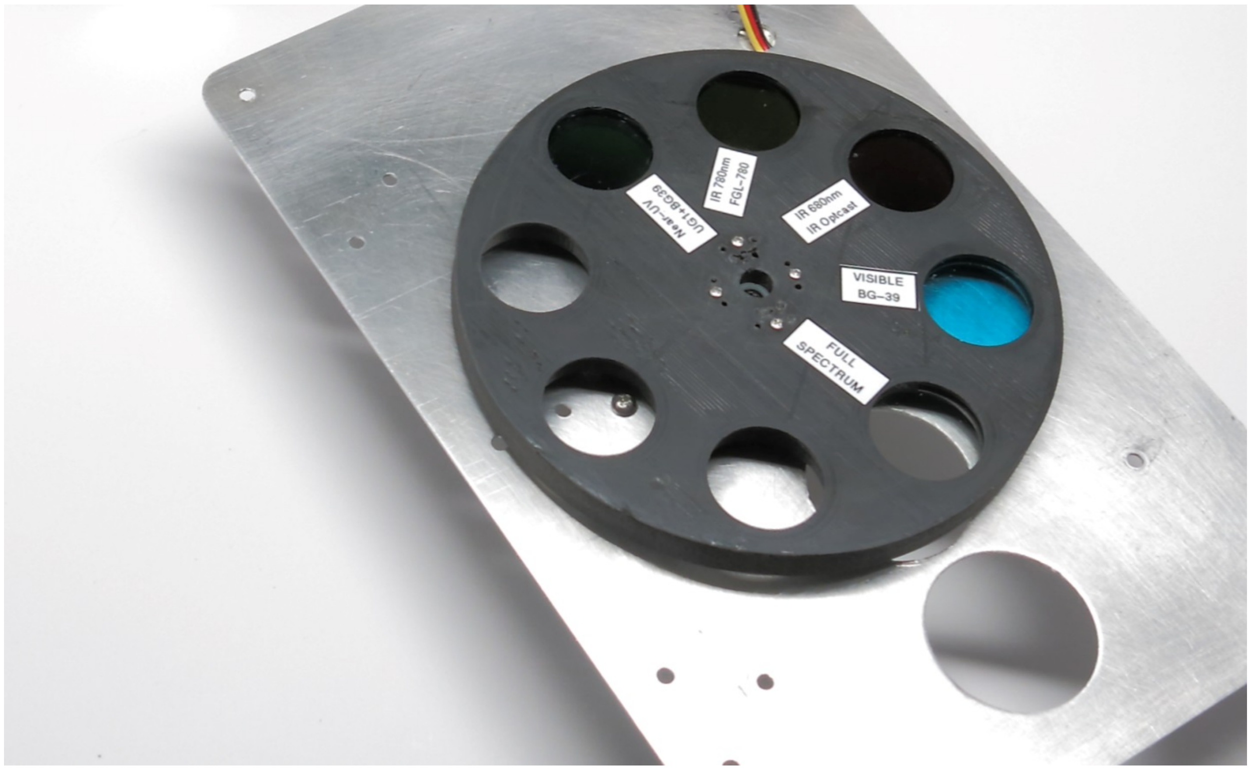


Figure 70 – Band selection filter wheel assembly showing filter placement.

Final Assembly

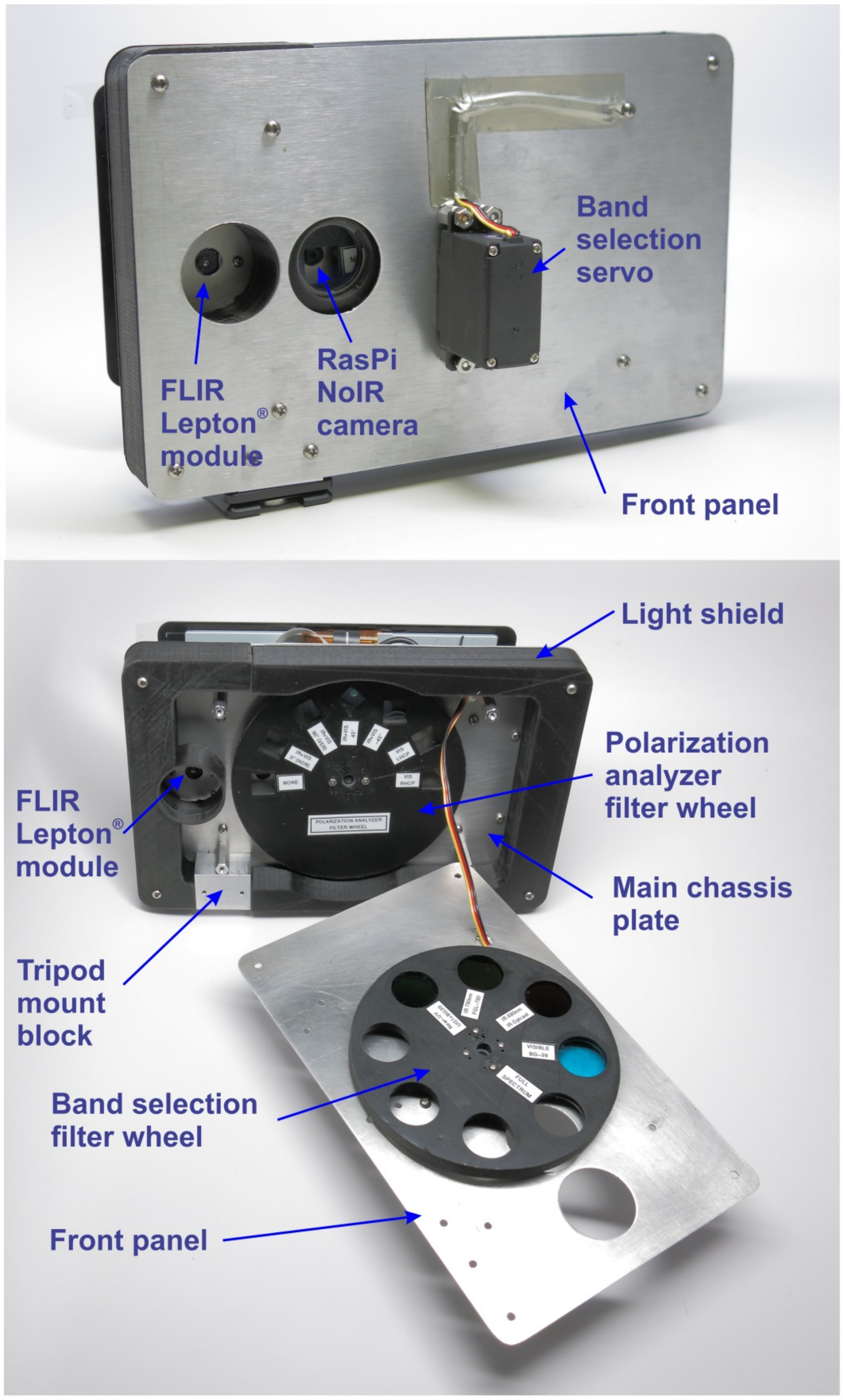


Figure 71 – Front assembly of DOLPi-UI showing the location of the filter wheels and camera modules.

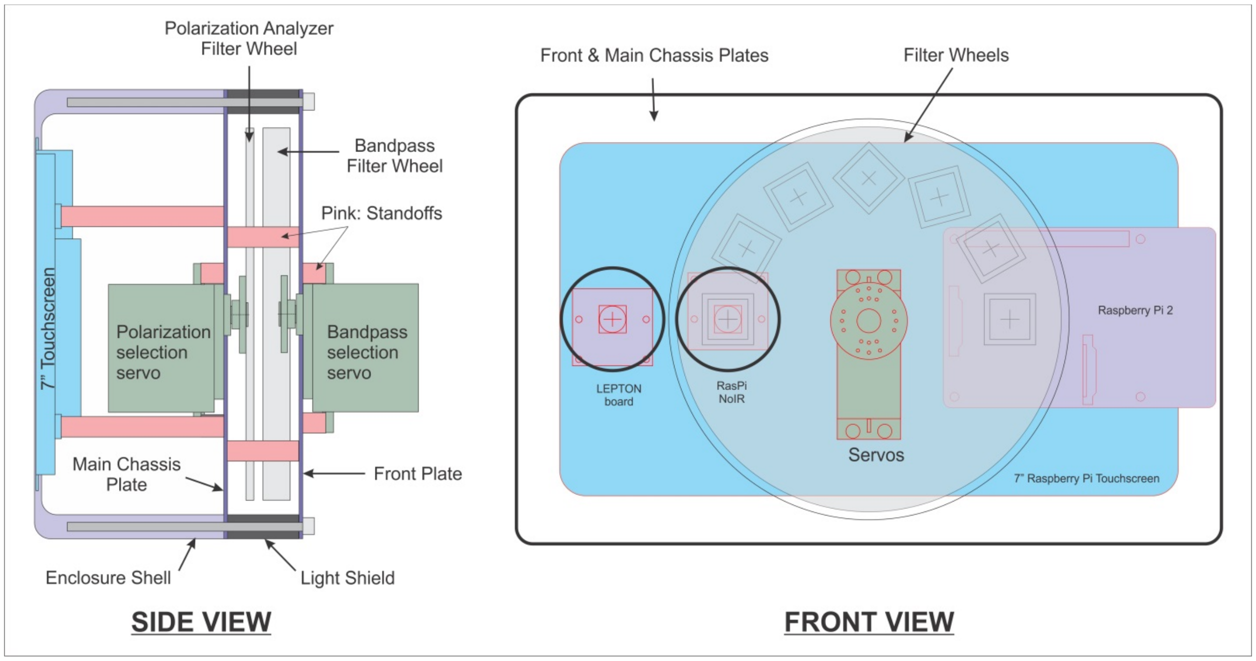


Figure 72 – DOLPi-UI's simplified assembly view

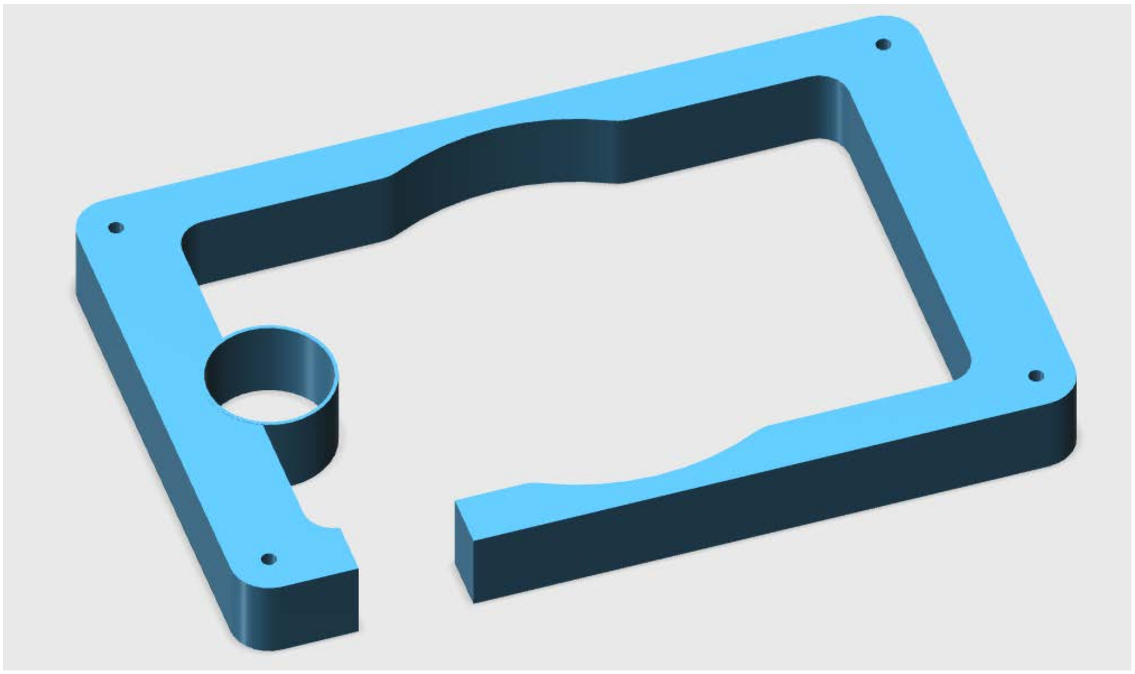


Figure 73 - Solid model rendering of LightShieldFrame.stl

1. 3D-print LightShieldFrame.stl using your printer's high-resolution mode. Figure 73 shows a rendering of the Light Shield.

2. 3D-print EnclosureShell.stl using your printer's high-resolution mode.
3. Screw a 4-40 nut to each of the 2" standoffs.
4. Screw the standoffs (with screw stub remaining after nut is in place) to Touchscreen. The touchscreen uses metric holes, but the 4-40 screws will retain it appropriately.
5. Machine the Tripod Mounting Block from a 7/8" x 7/8" profile bar of Multipurpose 6061 Aluminum. Tap tripod mount hole for ¼" screw. Tap holes to mount block to panels with a 4-40 tap.
6. Place the Tripod Mounting Block in its slot on the Light Shield.
7. Screw the Tripod Mounting Block to the Main Chassis Plate.
8. Screw the Touchscreen standoffs to the Main Chassis Plate.
9. Place the Light Shield onto the Main Chassis Plate.
10. Screw the Front Plate to its standoffs and to the Tripod Mounting Block.
11. Complete the assembly by screwing the Enclosure Shell to the Front Panel.

Software Installation

1. Install drivers for the Raspberry Pi 7" Touchscreen as described in http://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display
2. Add "display_rotate=2" in /boot/config.txt to flip the display 180^o
3. Edit /usr/share/X11/xorg.conf.d/10-evdev.conf to invert the X and Y axes for the touchscreen interface as follows:

```
Section "InputClass"
Identifier "RPi Touch"
MatchDevicePath "/dev/input/event0"
Option "InvertX" "1"
Option "InvertY" "1"
EndSection
```

4. Install picamera by following the instructions at http://picamera.readthedocs.org/en/latest/index.html
5. Install cv2 (OpenCV) with bindings to Python 2.7 by following the instructions at http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/
6. Install numpy, matplotlib and scipy: $ sudo apt-get install python-numpy python-scipy python-matplotlib
7. Install the Adafruit Servo Hat by following the instructions at https://learn.adafruit.com/adafruit-16-channel-pwm-servo-hat-for-raspberry-pi/overview
8. Install the FLIR Lepton® drivers following the instructions at: https://learn.sparkfun.com/tutorials/flir-lepton-hookup-guide
9. Test the basic functionality of the DOLPi-UI using the Python 2.7 code of Table 10.
10. Download the latest version of the DOLPi-UI application software from https://github.com/prutchi/DOLPi/ and run it with Python 2.7.

Table 10 – DOLPi_UI_1.py is used to test the basic functionality of the DOLPi-UI camera.

```
#   DOLPi_UI_1.py
#
#   This Python program demonstrates the DOLPi_UI universal polarimetric camera.
#   Operating mode is linear polarization only in this demo.
#
#   A servo motor rotates a polarization filter wheel in front of the
#   Raspberry Pi camera.  A second servo rotates a band-selection filter wheel
#   An Adafruit PWM Servo HAT drives the servos.
#
#   Bandpass must be set manually by uncommenting/commenting appropriate line
#   Resolution must be set manually by uncommenting/commenting appropriate line
#
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                           (MIT, opensource.org/licenses/MIT)
#
#
# Import the necessary packages
# -----------------------------
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
from Adafruit_PWM_Servo_Driver import PWM
import numpy as np
import matplotlib.pyplot as plt

# Initialize the Adafruit PWM HAT using the default address
# --------------------------------------------------------
pwm = PWM(0x40)
pwm.setPWMFreq(60)   # Set PWM frequency to 60 Hz

# Define servo PWM values for filters
# -----------------------------------
# Servo PWM values for different polarization analyzer filter wheel positions
servoNone = 620      # PWM setting for open window
servo0=544          # PWM setting for 0 degree filter
servo90=468         # PWM setting for 90 degree filter
servo45=392         # PWM setting for 45 degree filter
servo135=316        # PWM setting for -45 degree (=135 degree) filter
servoLHCP=240       # PWM setting for LHCP filter
servoRHCP=164       # PWM setting for RHCP filter
# Servo PWM values for different bandpass filter wheel positions
servoFull=165       # Full banwidth
servoVIS=274        # Visible range (BG-39 glass)
servoIR1=383        # Near-IR 680nm opticast
servoIR2=492        # Near-IR 780nm FGL-780
```

```
servoUV=600          # IR-supressed near-UV (UG-1 + BG-39)


# Uncomment to test polarization analyzer servo positions
# -------------------------------------------------------
#pwm.setPWM(0, 0, servoNone)  # No filter
#time.sleep(1)
#pwm.setPWM(0, 0, servo0)     # Zero degree filter
#time.sleep(1)
#pwm.setPWM(0, 0, servo90)    # 90 degree filter
#time.sleep(1)
#pwm.setPWM(0, 0, servo45)    # 45 degree filter
#time.sleep(1)
#pwm.setPWM(0, 0, servo135)   # -45 degree filter
#time.sleep(1)
#pwm.setPWM(0, 0, servoLHCP)  # LHCP filter
#time.sleep(1)
#pwm.setPWM(0, 0, servoRHCP)  # RHCP filter
#time.sleep(1)

# Uncomment to test bandpass servo positions
# ------------------------------------------
#pwm.setPWM(1, 0, servoFull) # Full bandwidth of Raspberry NoIR camera
#time.sleep(1)
#pwm.setPWM(1, 0, servoVIS)  # Visible range
#time.sleep(1)
#pwm.setPWM(1, 0, servoIR1)  # Near-IR >680nm
#time.sleep(1)
#pwm.setPWM(1, 0, servoIR2)  # Near-IR >780nm
#time.sleep(1)
#pwm.setPWM(1, 0, servoUV)   # Near-UV
#time.sleep(1)

# Select Bandpass
# ---------------
#pwm.setPWM(1, 0, servoFull)  # Full bandwidth of Raspberry NoIR camera
pwm.setPWM(1, 0, servoVIS)  # Visible range
#pwm.setPWM(1, 0, servoIR1)  # Near-IR >680nm
#pwm.setPWM(1, 0, servoIR2)  # Near-IR >780nm
#pwm.setPWM(1, 0, servoUV)   # Near-UV
time.sleep(1)

#Raspberry Pi Camera Initialization
#----------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
camera.resolution = (320, 240)
#camera.resolution = (640, 480)
```

```
#camera.resolution = (1280,720)
camera.framerate=30
rawCapture = PiRGBArray(camera)
camera.led=False

# Auto-Exposure Lock
# ------------------
pwm.setPWM(0, 0, servoNone)  #Ser filter wheel desired filter
time.sleep(0.1)    #Wait for filter wheel to move
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain

# Initialize flags
# ----------------
loop=True   #Initial state of loop flag
video=False #Use video port?  Video is faster, but image quality is significantly
           #lower than using still-image capture

# Main imaging loop
# -----------------
while loop:
    #grab an image from the camera with linear polarizer at 0 degrees
    pwm.setPWM(0, 0, servo0)
    time.sleep(0.2)    #Wait for filter wheel to move
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image0=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)

    #grab an image from the camera with linear polarizer at 90 degrees
    pwm.setPWM(0, 0, servo90)
    time.sleep(0.1)
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image90=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)

    #grab an image from the camera with linear polarizer at 45 degrees
    pwm.setPWM(0, 0, servo45)
    time.sleep(0.1)
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image45=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)
    #grab an image from the camera with linear polarizer at -45 degrees (=135 degrees)
    pwm.setPWM(0, 0, servo135)
    time.sleep(0.1)
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image135=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)

    #convert images to signed double (int16)
    image0_d=np.int16(image0)
    image90_d=np.int16(image90)
    image45_d=np.int16(image45)
    image135_d=np.int16(image135)
    #calculate Stokes parameters
    stokesI=image0_d+image90_d+.1
    stokesQ=image0_d-image90_d
    stokesU=image45_d-image135_d
    #calculate polarization parameters
    polInt=np.sqrt(1+np.square(.1+stokesQ)+np.square(.1+stokesU))  #Linear Polarization Intensity
    polDoLP=polInt/stokesI   #Degree of Linear Polarization
    polAoP=0.5*(np.arctan2(stokesU,stokesQ))  #Angle of Polarization

    #prepare DOLPi HSV image
    H=np.uint8((polAoP+(3.1416/2))*(180/3.1416))
    S=np.uint8(255*(polDoLP/np.amax(polDoLP)))
    V=np.uint8(255*(polInt/np.amax(polInt)))

    #prepare DOLPi RGB image
    R=image0
    B=image90
    G=image45
    imageDOLPi=cv2.merge([B,G,R])
    cv2.imshow("Image_DOLPi",imageDOLPi)  #Display DOLPi preview image

    imageDOLPiHSV=cv2.merge([H,S,V])
    DOLPiHSVinBGR=cv2.cvtColor(imageDOLPiHSV,cv2.COLOR_HSV2BGR)
    cv2.imshow("DOLPi_HSV",DOLPiHSVinBGR) #Display DOLPi HSV image

    k = cv2.waitKey(20)  #Check keyboard for input
    if k == ord('x'):   # wait for x key to exit
        loop=False


#  Prepare to leave
#  ----------------
pwm.setPWM(0, 0, servoNone) #return polarizer filter wheel to no-filter position
pwm.setPWM(1, 0, servoFull) #return bandwidth filter wheel to full bandwidth position
cv2.destroyAllWindows()
quit
```

Appendix V - DOLPi-EO Construction

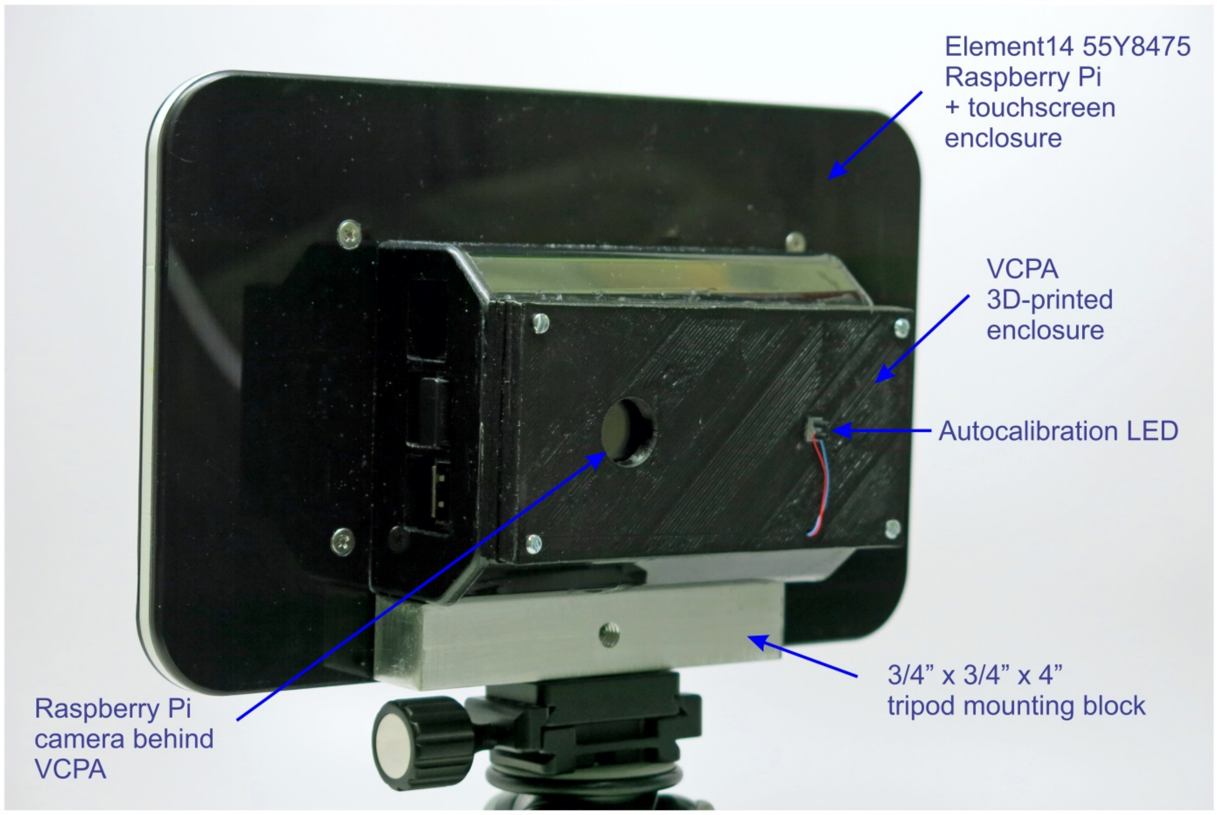


Figure 74 - Prototype for a productized version of the electro-optical version of DOLPi. The display is Raspberry Pi Foundation's new 7" touchscreen, and the main enclosure is a modified Element14 55Y8475 case.

As shown in Figure 74, I developed one last prototype for a productized version of the electro-optical version of DOLPi. Unfortunately, I had some issues yielding printed circuit boards, so the final prototype still has the circuit built on perfboard. This also translated into the need for a larger enclosure than would be needed had I used a PCB with surface-mount components.

On the positive side however, I'm very pleased with the elegance and performance of the new AC driver circuit which I believe is ready for final PCB layout into a RASPi hat. As shown in the block diagram of Figure 75, the driver is implemented trough an analog switch IC configured as an H-bridge that is constantly switching the polarity across the liquid crystal panel at around 2kHz. The drive voltage is derived directly from the D/A converter. The autocalibration circuit is the same as the one that I developed for the original DOLPi camera.

The schematic diagram for the circuit in the advanced DOLPi-EO prototype is shown in Figure 76. The PCB would include this circuit, as well as the reference designs for the ADS1015 and MCP4725 implemented in the Adafruit breakout boards.

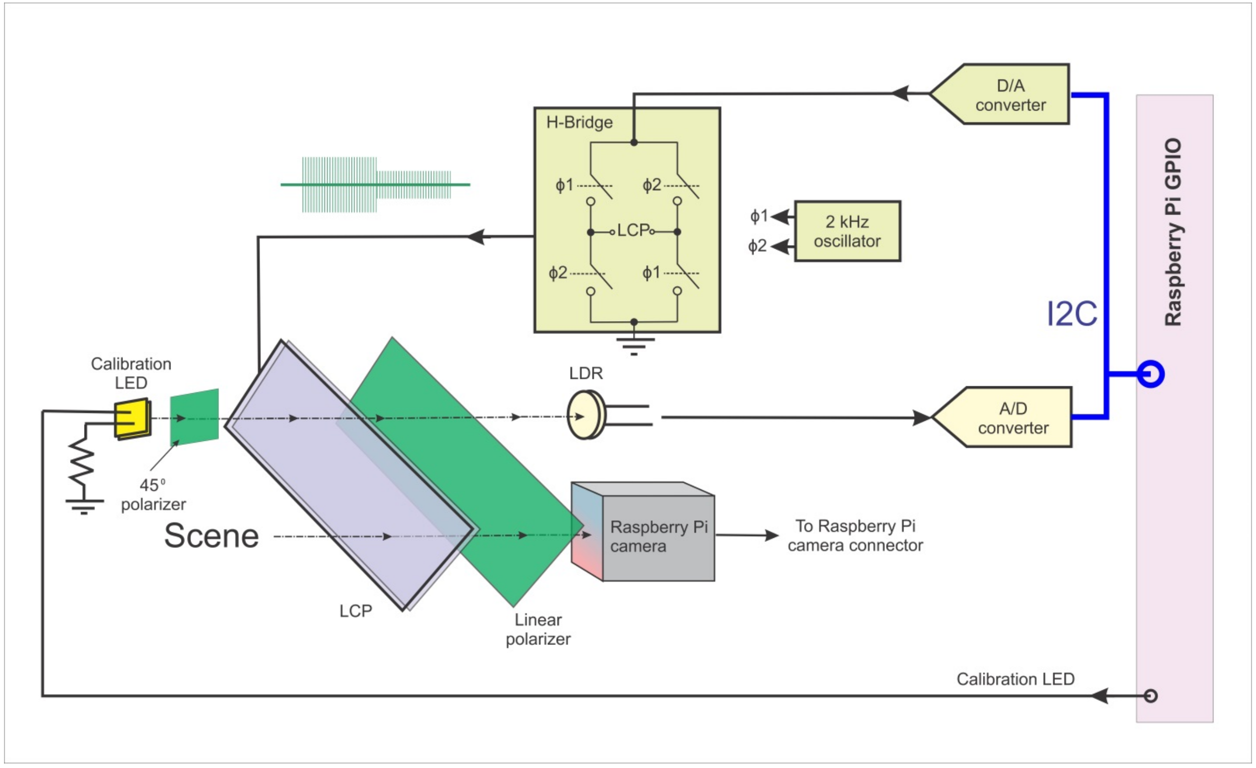


Figure 75 – Block diagram of the productized DOLPi-EO polarimetric camera. The VCPA is AC-driven at 2kHz. The drive amplitude is controlled from software via a D/A converter. The drive amplitude for 45° polarization shift is determined by an auto-calibration circuit consisting of an A/D converter that measures the intensity of 45° polarized light that goes through the VCPA as a function of D/A output.

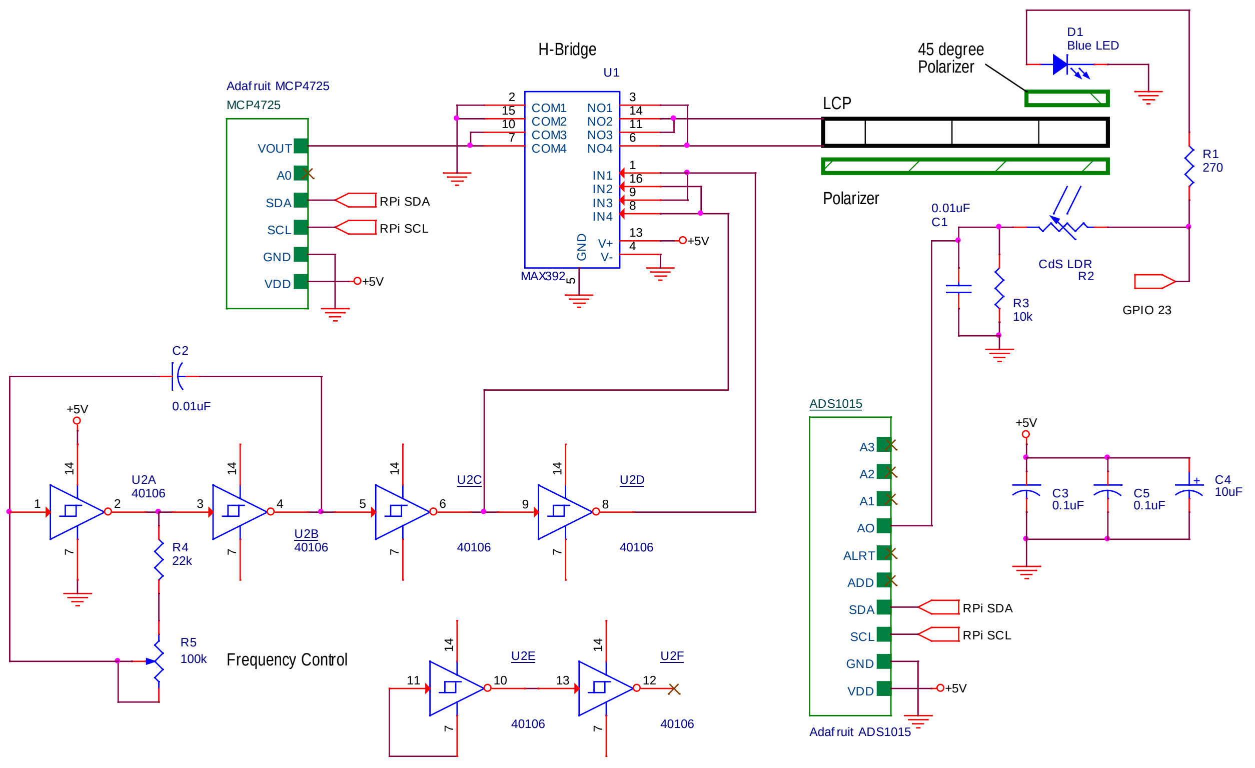


Figure 76 – Schematic diagram for the LCP driver and autocalibration circuit for the DOLPi-EO productized electro-optical polarimetric camera.

Table 11 - Bill of Materials for DOLPi-UI

Component reference	Component/Material	Source
RasPi	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	Adafruit ID:2358
SD Card	SanDisk SAEMSD64GBU3 64GB Extreme UHS-I microSDXC Memory Card (U3/Class 10)	B&H Photo
+5V Power Supply	5V 2A Switching Power Supply w/ 20AWG 6' MicroUSB Cable	Adafruit ID: 1995
RasPi Camera	Raspberry Pi Camera Board	Adafruit ID:1367
RasPi Touchscreen	Pi Foundation PiTFT - 7" Touchscreen Display for Raspberry Pi	Adafruit ID: 2718
Keyboard/Mouse	USB keyboard and mouse	Any suitable keyboard and mouse Preferred: Wireless Keyboard and Mouse Combo, Adafruit ID: 1738
D1	Blue LED (rectangular, ground to 4mm height)	Radio Shack 276-0013
R1	270 Ω, ¼ W	Any electronic parts store
R2	7mm diameter CdS LDR	Any electronic parts store
R3	10 kΩ, ¼ W	Any electronic parts store

R4	22 kΩ, ¼ W	Any electronic parts store
R5	100 kΩ trimmer potentiometer	Any electronic parts store
C1, C2	0.01 µF monolithic	Any electronic parts store
C3, C5	0.1 µF monolithic	Any electronic parts store
C4	10 µF / 16V tantalum	Any electronic parts store
Perfboard (will eventually be PCB)	0.1” x 0.1” pefboard cut to 98 mm x 55 mm	Any electronic parts store
GPIO Header	Extra-tall stacking 40 pin header	Adafruit ID: 19179
U1	MAX392 or similar low-impedance N.O. quad analog switch	Mouser Electronics Part No. 700-MAX392CPE
U2	CD40106 Hex Schmitt Trigger Inverters	Mouser Electronics Part No.595-CD40106BE
VCPA	VCPA hacked from auto-darkening welding mask filter per text	Online (Amazon, eBay, etc). Filter manufactured by “Mask” www.auto-mask.com (see Figure 13)
ADC	ADS1015 Breakout Board - 12-Bit ADC - 4 Channel with Programmable Gain Amplifier	Adafruit ID:1083
DAC	MCP4725 Breakout Board - 12-Bit DAC w/I2C Interface	Adafruit ID:935
Mechanical		
LCP Enclosure	3D-printed enclosure	3D-print DOLPi3.stl
Enclosure	Raspberry Pi Touchscreen Enclosure; For Use With:Raspberry Pi 7"Touch Screen with Pi B+ or Pi 2 Boards; External Height: 131mm; External Width: 213.2mm; External Depth: 51.5mm;	Element14 Catalog No. 55Y8475
Camera mounting screws	Four M2.5 x 1” nylon machine screws and eight matching nylon nuts	Hardware store
Tripod mounting block	¾” x ¾” x 4” aluminum machined as shown	Machined from McMaster-Carr 9008K12

DOLPi-EO Circuit Board Assembly

1. Cut a 0.1” x 0.1” pefboard to 98 mm x 55 mm
2. Notch the perfboard as shown in Figure 77 to allow passage of the camera’s ribbon cable.
3. Assemble the circuit shown in the schematic diagram of Figure 76 allowing necessary clearances for the various connectors on the Raspberry Pi 2.

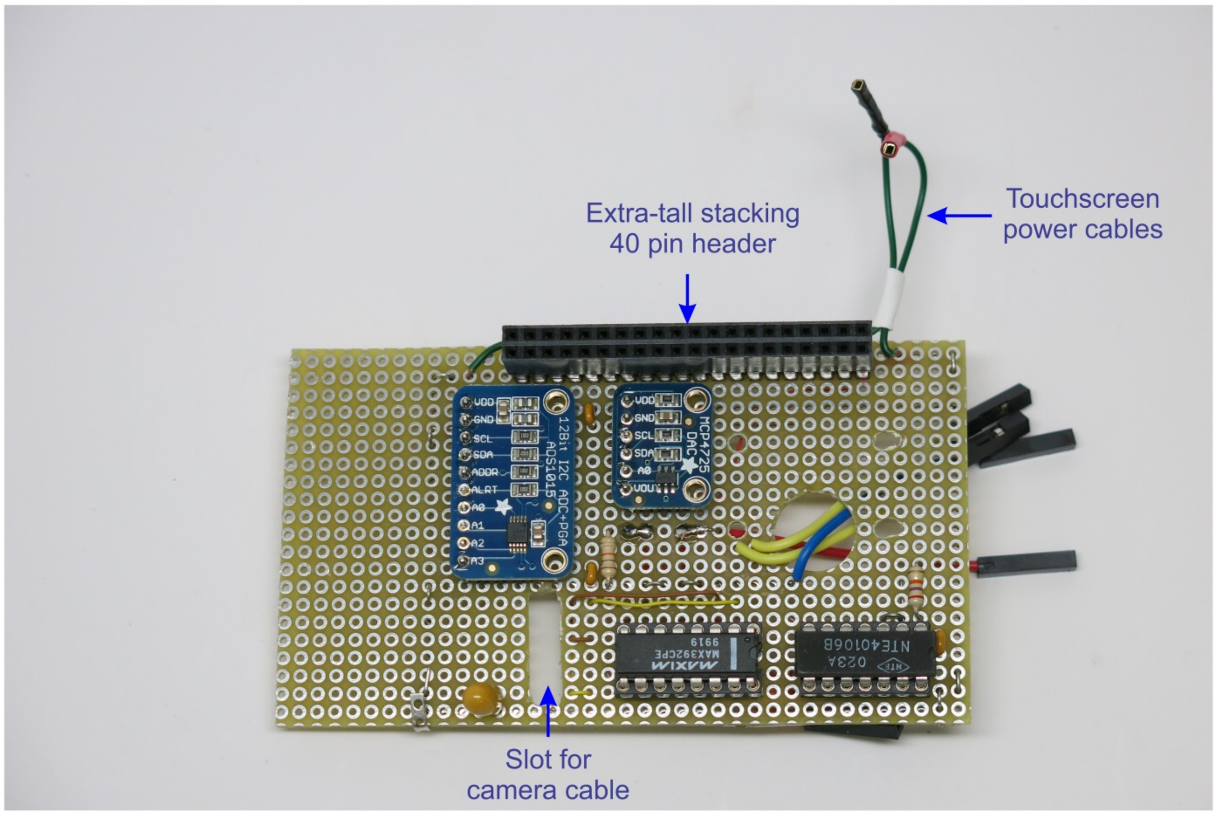


Figure 77 – Prototype board for the DOLPi-EO hat implements the circuit of Figure 76. The prototype board shows a cutout for the Raspberry Pi camera. However, the camera was not ultimately mounted straight to the circuit board because the VCPA could not be stacked inside the unmodified Element14 55Y8475 enclosure. This should be resolved once a PCB is used instead of a prototyping board.

DOLPi-EO VCPA Assembly

1. 3D-print the DOLPi3.stl file using your printer's high-resolution mode. Figure 78 shows a rendering of the VCPA enclosure's components.
2. Take apart the auto-darkening welding mask filter as shown in Figure 16. Opening the plastic enclosure is easy. The two half-shells forming the enclosure are glued together and can be separated with a blade or screwdriver. The LCP's drive wires can then be unsoldered (or cut) from the circuit.
3. Remove the purple glass filter from the LCP assembly by cutting the bit of transparent tape on its sides and lifting it with a sharp razor.
4. Remove one of the polarizer films. It peels off easily after lifting a corner with a sharp hobby knife. The remaining polarizer and LCP now form a voltage-controlled polarization analyzer. Please note that the polarizing film used in auto-darkening welding masks is oriented at 45^o with respect to the edges of the LCP. If you want the horizontal horizon to be the reference axis, replace the remaining polarizer by a piece of linear polarizing film oriented horizontally.

5. Tape a piece of linear polarizer oriented at 45^{o} (referenced to the polarization axis of the polarizer film of the prior step) on the opposite side of the LCP. This polarizer should cover only the area between the calibration LED and the LDR. It should not intrude the area of the camera.
6. Install the RasPi camera onto the 3D-printed enclosure main plate with nylon machine screws using two nuts as spacers.
7. Glue the LDR in place on the main plate.
8. Glue the LED in place on the cover plate.
9. Install the VCPA in place. It is recommended to tack it with two small pieces of tape.
10. Cut the back of the touchscreen enclosure to allow cables to go through.
11. Place and glue round-base nuts on the mounting plate.
12. Glue the mounting plate to the back of the touchscreen enclosure.

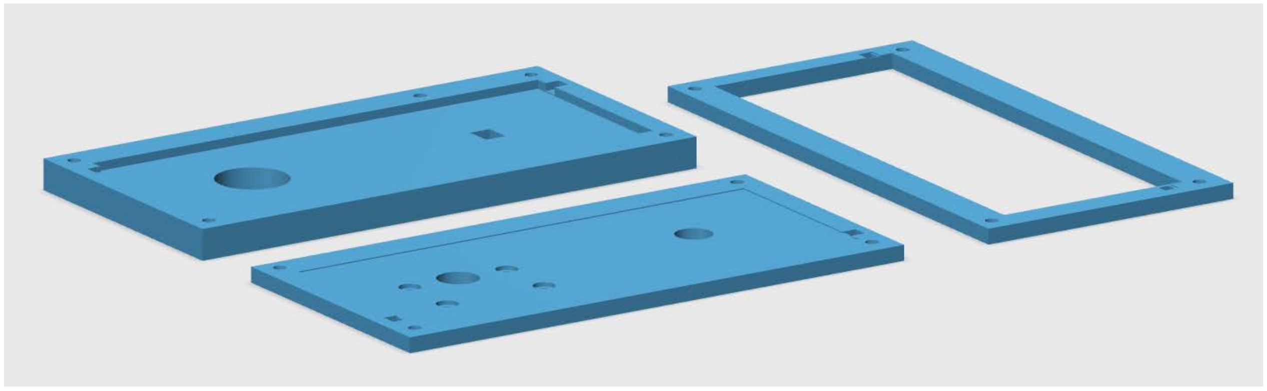


Figure 78 – Solid model of the VCPA assembly enclosure to be 3D-printed for the DOLPi-EO polarimetric camera.

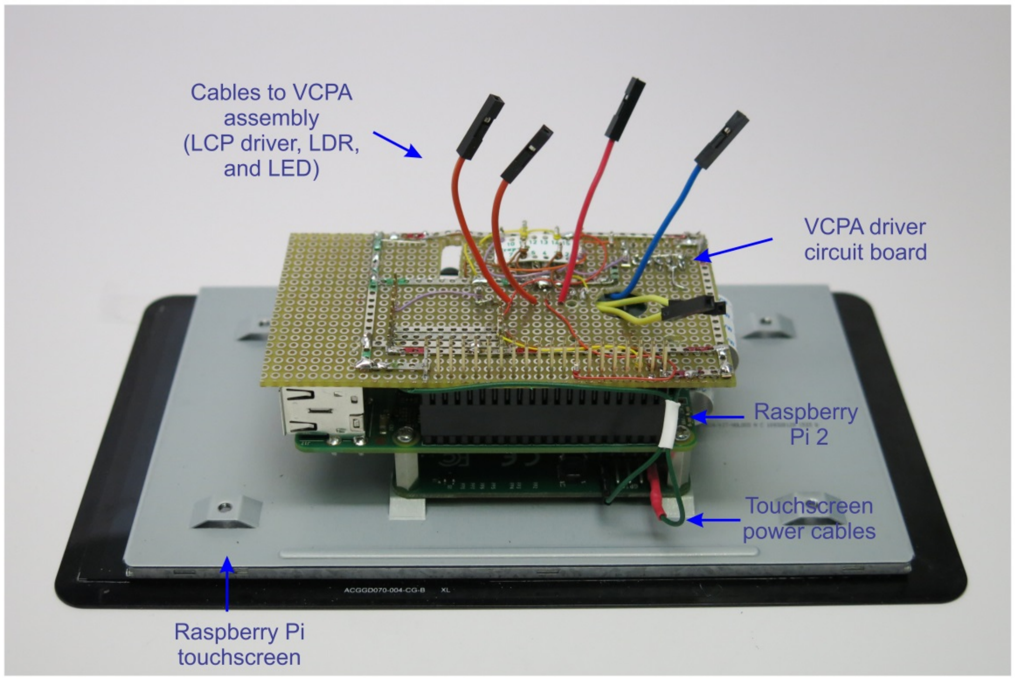

Cables to VCPA assembly (LCP driver, LDR, and LED)
VCPA driver circuit board
Raspberry Pi 2
Touchscreen power cables
Raspberry Pi touchscreen

Figure 79 – The DOLPi-EO circuit board is mounted as hat onto the RasPi's GPIO connector. Power for the touchscreen is brought to the LCD driver board from the hat.

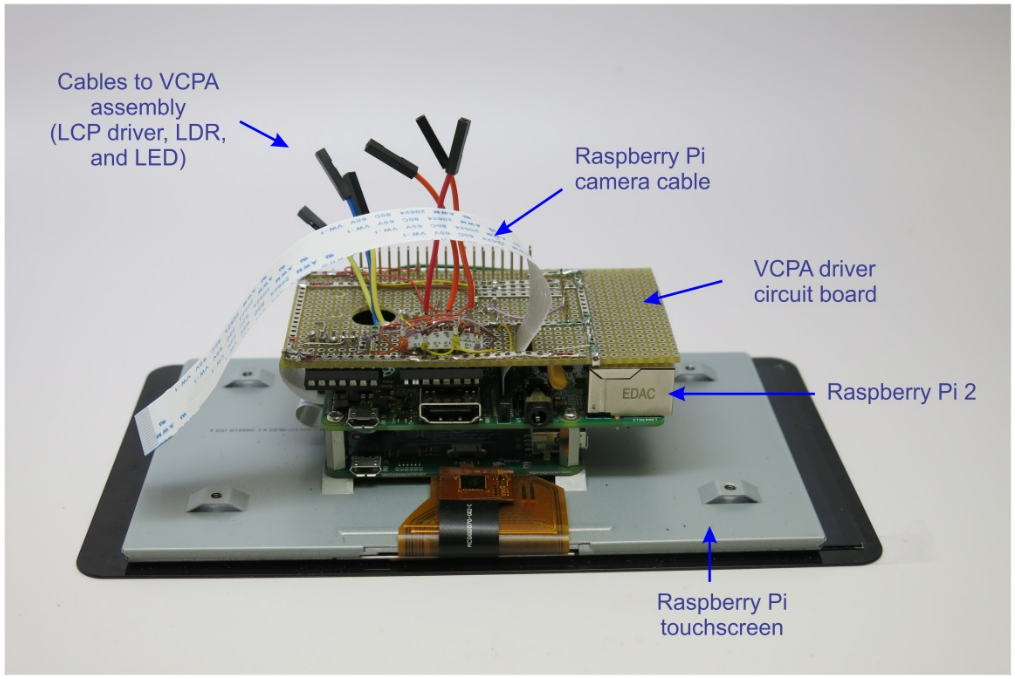

Cables to VCPA assembly (LCP driver, LDR, and LED)
Raspberry Pi camera cable
VCPA driver circuit board
Raspberry Pi 2
Raspberry Pi touchscreen

Figure 80 – The Raspberry Pi camera's cable is routed through a slot in the DOLPi-EO hat.

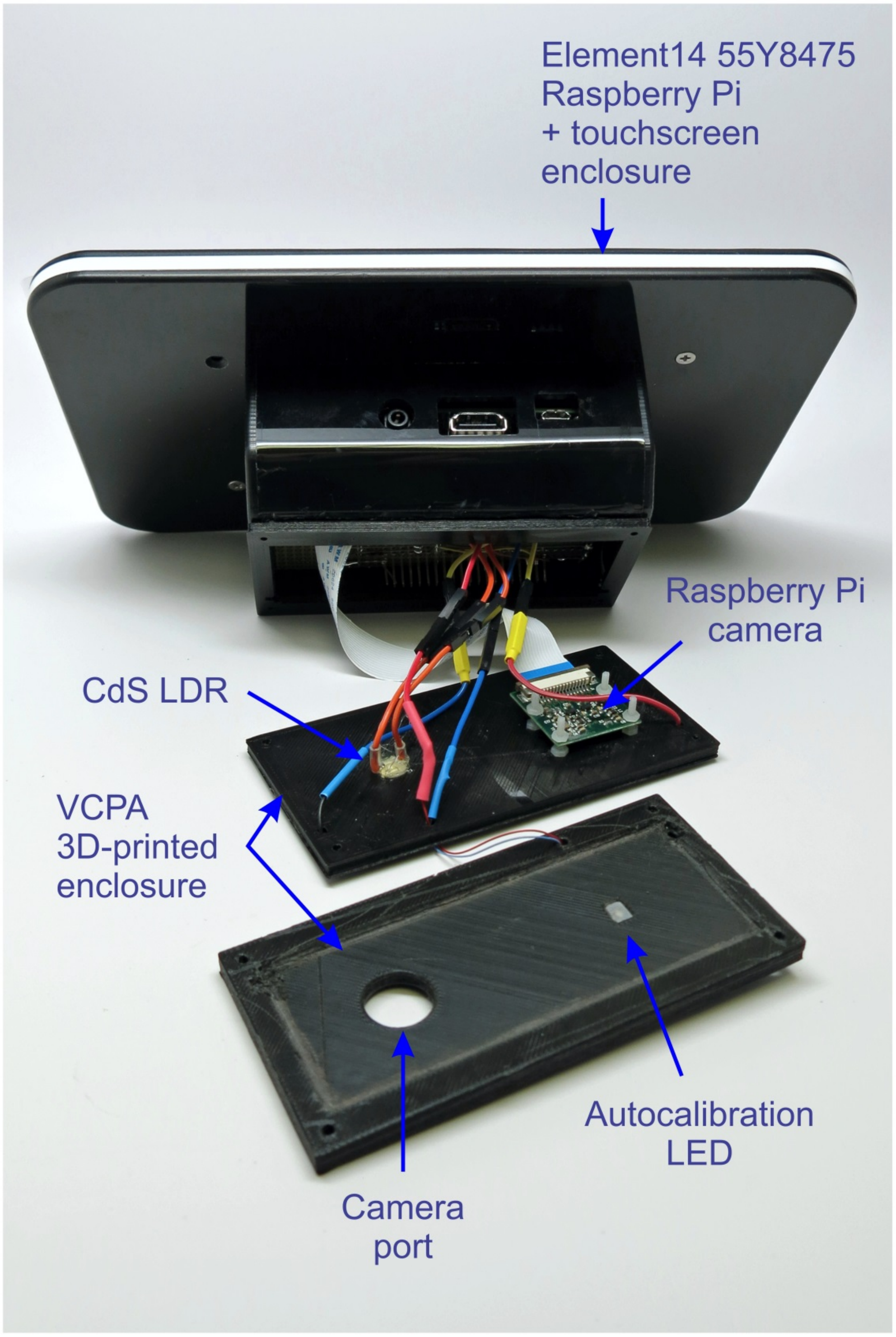


Figure 81 – The VCPA enclosure is 3D-printed. It holds the LCP with its polarizing filters, the Raspberry Pi camera, the autocalibration LED, and CdS LDR. The VCPA assembly is mounted to a 3D-printed frame glued to the carved-out back of the Element14 55Y8475 enclosure.

Final Assembly

1. Assemble the touchscreen and RasPi as shown in http://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display
2. Assemble the DOLPi-EO circuit board as a hat to the Raspberry pi.
3. Connect the camera to the Raspberry Pi.
4. Connect the LCP, LED, and LDR to the DOLPi-EO circuit board.
5. Screw the VCPA enclosure to the base plate to complete the assembly.

Software Installation

I installed all packages to run under Python 2.7. I chose not to use Python 3.x because many of the smaller scientific packages for computer vision, machine learning, and data science have not been migrated yet from Python 2.7. My code and instructions are thus written for Python 2.7, but there shouldn't be anything major that would prevent DOLPi-EO from running under Python 3.x, maybe with a few tweaks.

11. Install drivers for the Raspberry Pi 7" Touchscreen as described in http://www.element14.com/community/docs/DOC-78156/l/raspberry-pi-7-touchscreen-display
12. Add "display_rotate=2" in /boot/config.txt to flip the display 180^o
13. Edit /usr/share/X11/xorg.conf.d/10-evdev.conf to invert the X and Y axes for the touchscreen interface as follows:
```
Section "InputClass"
Identifier "RPi Touch"
MatchDevicePath "/dev/input/event0"
Option "InvertX" "1"
Option "InvertY" "1"
EndSection
```
14. Install picamera by following the instructions at http://picamera.readthedocs.org/en/latest/index.html
15. Install cv2 (OpenCV) with bindings to Python 2.7 by following the instructions at http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/
16. Install numpy, matplotlib and scipy: $ sudo apt-get install python-numpy python-scipy python-matplotlib
17. Install Adafruit_MCP4725: https://github.com/adafruit/Adafruit_MCP4725/
18. Install Adafruit_ ADS1X15: https://github.com/adafruit/Adafruit_ADS1X15/
19. Test the basic functionality of the DOLPi-EO camera by running the Python 2.7 code presented in Table 12.
20. Download the latest version of the DOLPi-EO application software from https://github.com/prutchi/DOLPi/ and run it with Python 2.7.

Table 12 – DOLPiAuto4.py is used to test the basic functionality of the DOLPi-EO camera.

```
#   DOLPiAuto4.py
#
#   This Python program demonstrates the DOLPi polarimetric camera
#   under automatic control
#
#   (c) 2015 David Prutchi, Ph.D., licensed under MIT license
#                            (MIT, opensource.org/licenses/MIT)
#
#   This version uses a voltage-controlled polarization analyzer (VCPA) that
#   consists of a liquid crystal panel(LCP) and a polarizer film.  The polarizer
#   film is placed between the LCP and camera.
#   The VCPA is driven by a DAC-controlled AC driver to 3 states:
#   1. Highest DAC output of 5V for no rotation of polarization
#   2. Pseudo 45 degrees of rotation
#   3. Low DAC output for 90 degrees of rotation
#
#   An auto-calibration setup is used to determine the optimal voltages to
#   drive the VCPA.  The calibrator consists of a diffuse blue-light
#   LED illuminating the VCPA through a polarizer set at pseudo 45 degrees with
#   respect to the VCPA's polarizer.
#   Light transmission is detected by a LDR behind the VCPA and
#   measured by an ADS1015 Analog-to-Digital (ADC) converter IC
#
#import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import RPi.GPIO as GPIO
import numpy as np
from Adafruit_MCP4725 import MCP4725  #controls MCP4725 DAC
from Adafruit_ADS1x15 import ADS1x15  #controls ADS1015 ADC

#IO PINS
#-------
calLEDpin=23   #Calibration LED connected to pin 23
dc0degpin=22   #0 degree VCPA DC bias manual mode
dc45degpin=21  #45 degree VCPA DC bias manual mode
GPIO.setwarnings(False)  #Don't issue warning messages if channels are defined
GPIO.setmode(GPIO.BCM)
GPIO.setup(calLEDpin,GPIO.OUT)
GPIO.setup(dc0degpin,GPIO.OUT)
GPIO.setup(dc45degpin,GPIO.OUT)
GPIO.output(calLEDpin,False)  #Turn calibration LED off
GPIO.output(dc0degpin,False)
GPIO.output(dc45degpin,False)
```

```
#ADS1015 A/D DEFINITIONS
#----------------------
# Select the A/D chip
ADS1015 = 0x00  # 12-bit ADC
#ADS1115 = 0x01          # 16-bit ADC

# Select the gain
#gain = 6144  # +/- 6.144V
gain = 4096  # +/- 4.096V
# gain = 2048  # +/- 2.048V
# gain = 1024  # +/- 1.024V
# gain = 512   # +/- 0.512V
#gain = 256   # +/- 0.256V

# Select the sample rate
#sps = 8    # 8 samples per second
#sps = 16   # 16 samples per second
#sps = 32   # 32 samples per second
#sps = 64   # 64 samples per second
#sps = 128  # 128 samples per second
#sps = 250  # 250 samples per second
#sps = 475  # 475 samples per second
sps = 860  # 860 samples per second

# Initialize the ADC using the default mode (use default I2C address)
# Set this to ADS1015 or ADS1115 depending on the ADC you are using!
adc = ADS1x15(ic=ADS1015)

#MCP4725 D/A DEFINITIONS
#----------------------
# Initialize the DAC using the default address
dac = MCP4725(0x62)
# DAC output connected to frontLCD
dac.setVoltage(0)

def cal():
   #DRIVE VOLTAGE CALIBRATION
   #-------------------------
   # A/D converter is connected to a CdS Light Dependent Resistor (LDR) placed
   # behind the VCPA.  A LED illuminates the VCPA through a piece of polarizer film.
   # This function finds the drive voltages at which the VCPA should be driven
   # to analyze at pseudo 45 degrees
   #
   import matplotlib.pyplot as plt     #Needed only if plotting
   print "Please wait while calibrating..."
   #
   GPIO.output(calLEDpin,True)        #Turn calibration LED on
```

```
    #
    # Initialize LCP and flush A/D
    for i in range (1,5):
        dac.setVoltage(4095)          #Turn LCP full ON
        time.sleep(.1)                #Let it settle
        dac.setVoltage(0)             #Turn LCP OFF
        flush=adc.readADCSingleEnded(0, gain, sps) #Flush the A/D
    #
    #Initialize the lists to hold the graph
    vol=[]                          #List to hold DAC voltage codes
    light=[]                        #List to hold light transmission
    for volt in range(0,4095,10):      #Iterate through possible VCPA voltage values (steps of 10 are OK)
        dac.setVoltage(volt)          #Apply voltage to VCPA
        time.sleep(0.05)              #Let it settle
        vol.append(volt)              #Add voltage code to list
        # Read channel 0 in single-ended mode using the settings above
        light.append(adc.readADCSingleEnded(0, gain, sps)) # Measure light level
        #time.sleep(0.15)             #Wait before next step
    #Leave loop with LCPand LED off
    GPIO.output(calLEDpin,False)       #Turn calibration LED off
    dac.setVoltage(0)                 #Turn LCP off

    #
    plt.plot(vol,light)               #Plot VCPA transmission as function of voltage
    plt.xlabel('D/A voltage code')
    plt.ylabel('Light transmission A/D counts')
    plt.title('Light transmission through VCPA')
    plt.show()                        #Show the plot

    light_index_max=light.index(max(light)) #Find index of maximum transmission
    maxVCPAvolt=vol[light_index_max]       #Find DAC voltage code for maximum transmission
    print "max DAC code", maxVCPAvolt
    return  maxVCPAvolt
voltVCPA45=cal()
voltVCPA90=300
voltVCPA0=3000

#Raspberry Pi Camera Initialization
#----------------------------------
#Initialize the camera and grab a reference to the raw camera capture
camera = PiCamera()
#Select camera resolution
#-----------------------
camera.resolution = (320, 240)
#camera.resolution = (640, 480)
#camera.resolution = (1280,720)
camera.framerate=30
rawCapture = PiRGBArray(camera)
```

```
camera.led=False
# Flip camera if necessary
# ----------------------
camera.hflip=True
camera.vflip=True

#Auto-Exposure Lock
#-----------------
# Wait for the automatic gain control to settle
time.sleep(2)
# Now fix the values
camera.shutter_speed = camera.exposure_speed
camera.exposure_mode = 'off'
gain = camera.awb_gains
camera.awb_mode = 'off'
camera.awb_gains = gain

#Initialize flags
#---------------
loop=True  #Initial state of loop flag
first=True #Flag to skip display during first loop
video=True #Use video port?  Video is faster, but image quality is significantly
           #lower than using still-image capture

#Main imaging loop
#----------------
while loop:
   #grab an image from the camera at 0 degrees
   dac.setVoltage(0)
   dac.setVoltage(voltVCPA0)
   GPIO.output(dc0degpin,True)
   GPIO.output(dc45degpin,False)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image0=rawCapture.array[:,:,1]
   #image0=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)

   #grab an image from the camera at pseudo 45 degrees
   dac.setVoltage(0)
   dac.setVoltage(voltVCPA45)
   GPIO.output(dc0degpin,False)
   GPIO.output(dc45degpin,True)
   time.sleep(0.05)
   rawCapture.truncate(0)
   camera.capture(rawCapture, format="bgr",use_video_port=video)
   image45=rawCapture.array[:,:,1]
   #image45=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)
```

```
    #grab an image from the camera at 90 degrees
    dac.setVoltage(0)
    dac.setVoltage(voltVCPA90)
    GPIO.output(dc0degpin,False)
    GPIO.output(dc45degpin,False)
    time.sleep(0.05)
    rawCapture.truncate(0)
    camera.capture(rawCapture, format="bgr",use_video_port=video)
    image90=rawCapture.array[:,:,1]
    #image90=cv2.cvtColor(rawCapture.array,cv2.COLOR_BGR2GRAY)

    #convert images to signed double (int16)
    image0_d=np.int16(image0)
    image90_d=np.int16(image90)
    image45_d=np.int16(image45)
    #calculate Stokes parameters
    stokesI=image0_d+image90_d+1
    stokesQ=image0_d-image90_d
    stokesU=(2*image45_d)-stokesI
    #calculate polarization parameters
    polInt=np.sqrt(1+np.square(stokesQ+.1)+np.square(stokesU+.1))  #Linear Polarization Intensity
    polDoLP=polInt/stokesI   #Degree of Linear Polarization
    polAoP=0.5*(np.arctan2(stokesU,stokesQ))  #Angle of Polarization

    #prepare DOLPi HSV image
    H=np.uint8((polAoP+(3.1416/2))*(180/3.1416))
    S=np.uint8(255*(polDoLP/np.amax(polDoLP)))
    V=np.uint8(255*(polInt/np.amax(polInt)))
    imageDOLPiHSV=cv2.merge([H,S,V])
    DOLPiHSVinBGR=cv2.cvtColor(imageDOLPiHSV,cv2.COLOR_HSV2BGR)
    cv2.imshow("DOLPi_HSV",DOLPiHSVinBGR)

    #prepare DOLPi RGB image
    R=image0
    B=image90
    G=image45
    imageDOLPi=cv2.merge([B,G,R])
    cv2.imshow("Image_DOLPi",imageDOLPi)  #Display DOLPi preview image

    GPIO.output(dc0degpin,False)
    GPIO.output(dc45degpin,False)
    first=False
    k = cv2.waitKey(5)  #Check keyboard for input
    if k == ord('x'):   # wait for x key to exit
        loop=False

#Prepare to leave
```

```
#----------------

#Save desired images as .jpg files
#---------------------------------
#cv2.imwrite("image0.jpg",image0)
#cv2.imwrite("image90.jpg",image90)
#cv2.imwrite("image45.jpg",image45)
cv2.imwrite("RGBpol.jpg",imageDOLPi)
#cv2.imwrite("image0g.jpg",R)
#cv2.imwrite("image90g.jpg",G)
#cv2.imwrite("image45g.jpg",B)
cv2.imwrite("HSVpol.jpg",DOLPiHSVinBGR)

#Exit
#----
dac.setVoltage(0)                    #Turn LCP drive off
cv2.destroyAllWindows()
quit
```

Appendix VI – DOLPi-VISOR

As described in the section entitled “Productizing DOLPi”, I developed a final prototype suitable as the basis for manufacturing a simple direct-visualization polarization visor. When viewed through this visor, man-made objects appear to flash as shown in Figure 19.

This device is very inexpensive and can be used as an alternative aid for locating landmines and unexploded munitions. Of course, it has many other applications where polarized reflections need to be highlighted above natural backgrounds. The performance of this simple visor is quite impressive since neither camouflage nor moderate foliage stops it from highlighting man-made targets because adequate flashing can still be observed.

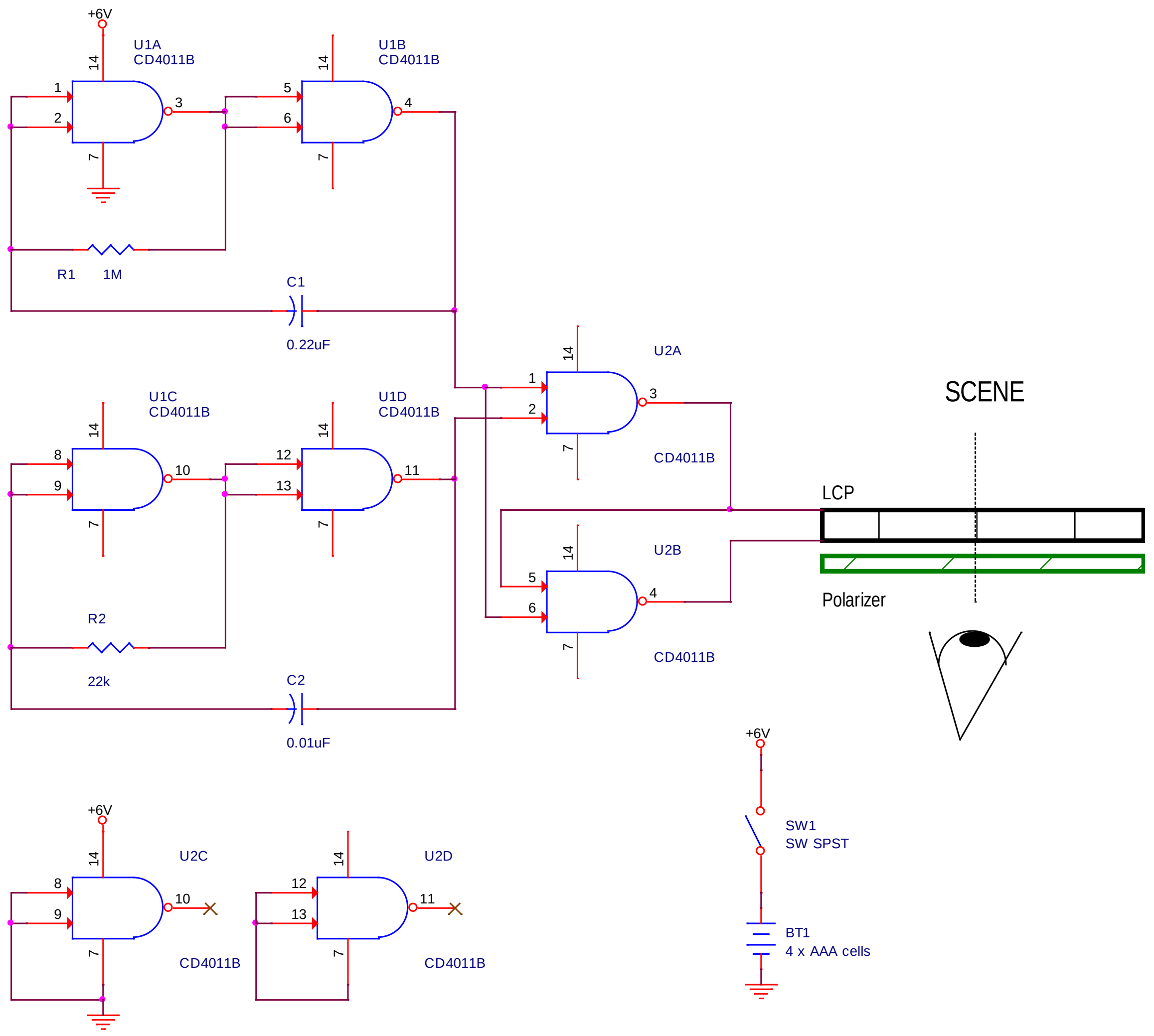


Figure 82 – Schematic diagram of the DOLPi VISOR. This is a self-contained 2-state polarization analyzer that implements a simple version of Boyd B. Bushman’s man-made object detector (see Figure 18). The oscillator built around U1A and U1B runs at around 3 Hz, while the oscillator built around U1C and U1D runs at around 1 kHz. The LCP is AC driven with a square wave with peak-to-peak amplitude of 12V in 167 ms bursts that repeat every 333 ms.

Table 13 - Bill of Materials for DOLPi-UI

Component reference	Component/Material	Source
U1, U2	CD4011BE	Mouser Electronics Part No. 595-CD4011BE
R1	1 MΩ ¼ W resistor	Any electronic parts store
R2	22 kΩ ¼ W resistor	Any electronic parts store
C1	0.22 µF capacitor	Any electronic parts store
C2	0.01 µF capacitor	Any electronic parts store
VCPA	VCPA hacked from auto-darkening welding mask filter per text	Online (Amazon, eBay, etc). Filter manufactured by "Mask" www.auto-mask.com (see Figure 13)
VCPA enclosure	Enclosure halves from auto-darkening welding mask filter per text	
Perfboard (will eventually be PCB)	0.1" x 0.1" pefboard cut to 2.75" x 7"	Any electronic parts store
Battery pack/switch (BT1/SW1)	RadioShack® Enclosed 4 "AAA" Battery Holder	RadioShack® Model #: 27-411
Cells for battery pack	Four alkaline AAA cells	Any electronic parts store
Miscellaneous	14-pin IC sockets, wire, hot glue, Velcro™, etc.	

The circuit shown in the schematic diagram of Figure 82 is straightforward. U1A, U1B and feedback resistor R1 and capacitor C1 form a square-wave oscillator that runs at around 3 Hz, while the oscillator built around U1C and U1D runs at around 1.5 kHz. These oscillators are ANDed by gates U2A and U2B to produce the drive signal for the LCP. The AC drive is a square wave with peak-to-peak amplitude of 12V and comes in 167 ms bursts that repeat every 333 ms to make polarized reflections viewed through the visor flash at 3Hz.

DOLPi-VISOR Assembly

1. Cut a piece of 0.1" x 0.1" pefboard to 2.75" x 7"
2. Carve out a window for the visor as shown in Figure 83.
3. Assemble the circuit shown in the schematic diagram of Figure 82.
4. Take apart the auto-darkening welding mask filter as shown in Figure 16. Opening the plastic enclosure is easy. The two half-shells forming the enclosure are glued together and can be separated with a blade or screwdriver. The LCP's drive wires can then be unsoldered (or cut) from the circuit. Keep the enclosure halves.
5. Remove the purple glass filter from the LCP assembly by cutting the bit of transparent tape on its sides and lifting it with a sharp razor.
6. Remove one of the polarizer films. It peels off easily after lifting a corner with a sharp hobby knife. The remaining polarizer and LCP now form a voltage-controlled polarization analyzer. Please note that the polarizing film used in auto-darkening welding masks is oriented at 45^o with respect to the edges of the LCP. If you want the horizontal horizon to be the reference axis, replace the remaining polarizer by a piece of linear polarizing film oriented horizontally.
7. Hot-glue the LCP enclosure as shown in Figure 83.

8. Connect the drive wires to the LCP and hot-glue it in place.
9. Insert four AAA cells into the battery pack.
10. Attach battery pack to circuit board using adhesive-backed Velcro™.
11. Test the device by turning it on and looking at a piece of polarizing film or LCD (e.g. computer LCD display, calculator, etc.). Make sure that the polarizer film side of the VCPA faces your eyes. The side of the LCP with no polarizer should face the scene (Figure 84).

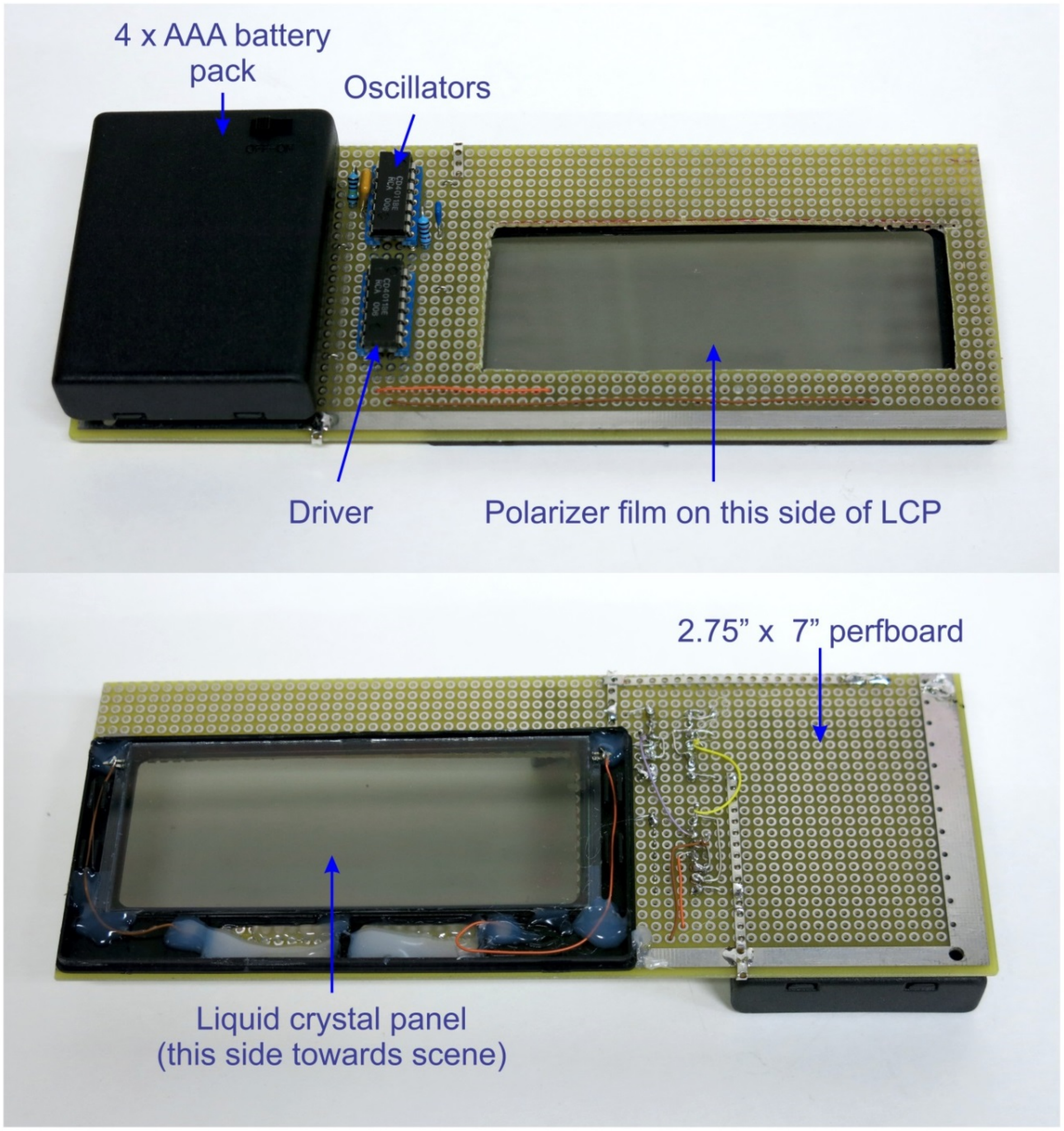

4 x AAA battery pack
Oscillators
Driver
Polarizer film on this side of LCP
2.75" x 7" perfboard
Liquid crystal panel (this side towards scene)

Figure 83 – Prototype of the productized direct-visualization DOLPi-VISOR. This simple, but useful device is constructed on a piece of perfboard (will eventually be a PCB) and is powered by four AAA cells. The top of the LCP enclosure is shown removed to show LCP connection and assembly detail.

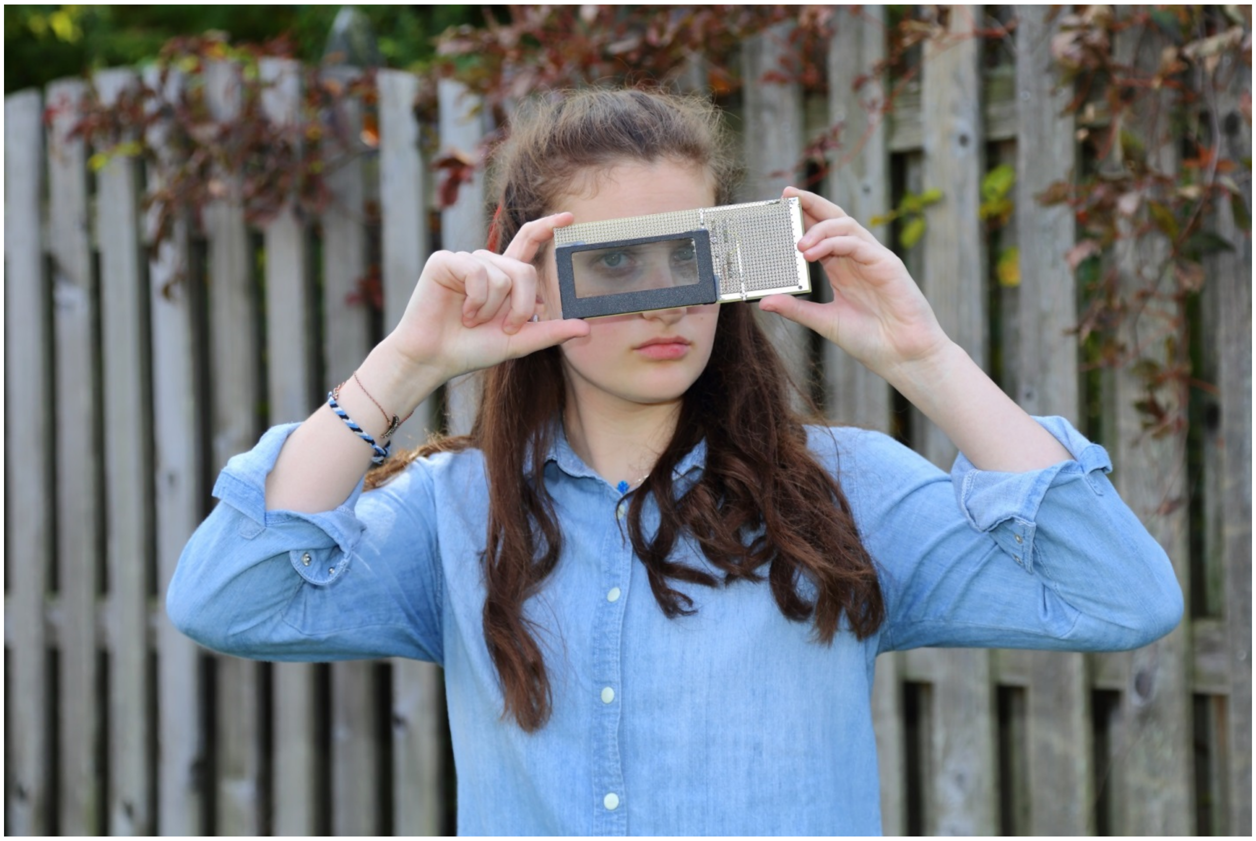


Figure 84 - To use the DOLPi VISOR you view the scene through the VCPA with the polarizer film facing you. This is my daughter Hannah demonstrating the use of the DOLPi VISOR.

About the Author

David Prutchi received his Ph.D. in Engineering from Tel-Aviv University in 1994, and then conducted post-doctoral research at Washington University. His area of expertise is the development of active implantable medical devices, and is currently the Vice President of Engineering at Impulse Dynamics. He is an adept do-it-yourselfer dedicated to bringing cutting-edge experimental physics within grasp of fellow science buffs.

David has published over thirty papers and holds over seventy patents. He is the lead author of the book "Design and Development of Medical Electronic Instrumentation - A Practical Perspective of the Design, Construction and Test of Medical Devices" and co-author of the book "Exploring Quantum Physics Through Hands-On Projects", both published by John Wiley & Sons. David's other off-hours interests include ham radio (extra-class license N2QG), radioastronomy, travel, and photography.

Contact via: contact <at> prutchi <dot> com

Please visit:

- www.prutchi.com
- www.diyPhysics.com
- www.implantable-device.com