

CAMS

Camera-experimenten - Voorbereidingsfase

Snelle camera's bieden de mogelijkheid om de positie van objecten als functie van de tijd nauwkeurig vast te leggen. Wanneer de beschrijving van de beweging volgens de wetten van de klassieke mechanica bekend is, kan er een kwantitatieve analyse worden gemaakt.

Met camerametingen kun je een reeks van experimenten doen. Essentieel bij experimenten met behulp van beeldanalyse zijn het goed onderscheid maken tussen het object en de achtergrond, een verstandig ontworpen experimentopzet en een goed gebruik van de mogelijkheden in Python voor *object tracking*.

Deze instructie is een beschrijving van de (opdrachten in) de voorbereidingsfase. Er is een aparte (korte) instructie voor de projectfase, en op Blackboard zijn veel voorbeelden en suggesties te vinden voor projecten.

Inleiding

Bij de camera-experimenten ga je een opname maken van een fysisch-mechanisch probleem. Het doel is om objecten in de beelden te herkennen en te volgen als functie van de tijd, en vervolgens de beweging van de beelden te koppelen aan een theoretische beschrijving.

De voorbereidingsfase betreft twee opdrachten, waarin stap voor stap een aantal voorbereidende acties worden doorlopen om objecten betrouwbaar en langdurig te kunnen volgen:

1. Bouw een eenvoudige opstelling, bedien de camera, installeer de benodigde (Python-)packages en leer hoe een object te identificeren.
2. Leer hoe een bewegend object (pingpongbal) in een filmpje te volgen, met basale en (later) geavanceerde beeldbewerkingsopties.

De deelopdrachten zijn in de instructie aangegeven met bullets (•). De opdrachten betreffen soms het uitvoeren van een (eenvoudig) experiment, en soms het maken of herschrijven van Python-code. Op Blackboard vind je de opnames die gebruikt zijn voor de voorbeelden in deze handleiding.

Rapportage in het labjournaal bestaat voor de voorbereidingsfase uit de code voor een bepaalde opdracht, de metingen die je gedaan hebt, en aanvullende observaties en uitwerkingen.

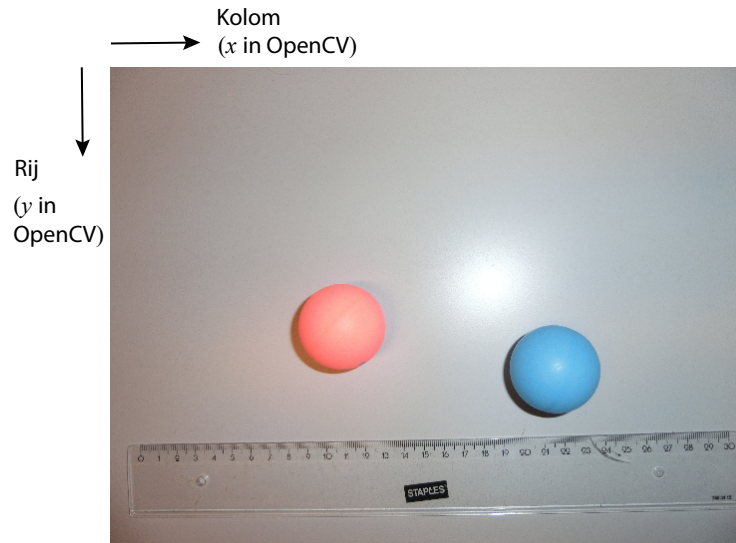
Bij de opzet van de tracking hebben wij gehad aan de website <https://www.pyimagesearch.com/>, waar nog veel meer mooie voorbeelden van beeldinterpretatie staan gedemonstreerd.

Camera

Bij de experimenten kun je werken met twee cameratypes in bezit van het practicum. De Exilim EX-ZR10 is een compacte camera met variabele opnamesnelheid of *framerate* (tot 1000 frames per seconde, fps, bij 224×64 pixel). De GoPro Hero 9 is een nieuwere camera die met hoge resolutie kan meten (1920×1080 pixel bij 240 fps), tot maximaal 240 fps.

Beide camera's werken erg intuïtief, dus we geven geen gedetailleerde bedieningsinstructies. Uitgebreide handleidingen voor beide camera's zijn te vinden op Blackboard.

► Tips zijn in de handleiding als volgt weergegeven.



Figuur 1: Voorbeeldframe met Exilim: twee pingpongballetjes met een liniaal om de werkelijke afstanden te iken. In het figuur is aangegeven welke richting in OpenCV correspondeert met x en welke met y .

1 Werken met camera en OpenCV

Deze eerste opdracht maken en bewerken we een enkele foto. In de tweede opdracht maken we de stap naar het eigenlijke volgen oftewel *tracken*. In de instructie noemen we een foto of een enkel shot uit een filmpje consequent een *frame*. Het deel van het beeld dat we willen volgen is vaak een object of een markering die erop is aangebracht (we zullen voor de eenvoud vanaf nu steeds spreken van het *object*).

1.1 Pingpongballen

In de voorbereidingsfase zijn de objecten die we bekijken pingpongballen. Het is in eerste instantie het eenvoudigst om met twee gekleurde pingpongballen te werken. Zie voor een voorbeeld Fig. 1.

1.2 OpenCV

Voor de beeldinterpretatie gebruiken we commando's uit het package OpenCV. Deze module is niet standaard geïnstalleerd met Anaconda. We leggen eerst uit hoe je die installatie uitvoert op je laptop (op de practicum-pc's is OpenCV al wel geïnstalleerd). Stap voor stap voor Windows:¹

¹Op Mac-systemen blijkt de installatie wat weerbarstiger vanwege problemen met afhankelijkheden tussen de verschillende Python-packages. Dit kan omzeild worden met een aparte 'environment' in Anaconda, waarbinnen alles wel op elkaar aansluit.

- Open de terminal via Applications | Utilities | Terminal
- Volg de stappen (typewriter font moet ingevoerd worden in de terminal):
 1. `conda update conda` (update naar de nieuwste versie van het conda package management system)
 2. `conda create -n opencv` (maak de nieuwe lege environment opencv)
 3. `conda activate opencv` (vanaf nu hebben al je acties betrekking op de environment opencv)
 4. `conda install -c anaconda spyder` (installeer spyder editor in de environment)
 5. `conda install -c anaconda matplotlib` (installeer package matplotlib); andere packages, bijvoorbeeld scipy, kunnen op een later moment op dezelfde manier geïnstalleerd worden.
 6. `conda install -c conda-forge opencv` (installeer package opencv, en daarbij allerlei andere packages zoals numpy)

1. Ga naar Start | All Programs | Anaconda3 | Anaconda Prompt . Rechtsklik en Selecteer Run as Administrator (het kan even duren voordat de prompt actief wordt)
2. Typ `conda install -c conda-forge opencv` . conda-forge is de verzameling packages waar opencv wordt opgehaald.
3. Je krijgt vervolgens de vraag of een aantal andere packages moeten worden bijgewerkt of overruled. Kies y. De packages worden nu gedownload en bijgewerkt (afhankelijk van de snelheid van de netwerkverbinding kan dit wel enkele minuten duren).
4. Met `exit` kun je de prompt weer verlaten.

Veel informatie over OpenCV kan gevonden worden in de online documentatie:

<https://docs.opencv.org/master/index.html>

Een frame kan ingeladen en bewerkt worden met commando's met voorvoegsel `cv2`. Zie Code 1

Code 1: Eerste gebruik van OpenCV.

```
# -*- coding: utf-8 -*-
"""
First usage of OpenCV
"""
# Load package OpenCV
import cv2

# Locations for example pictures
# Download and save to local directory
# Exilim
# https://nspracticum.science.uu.nl/DATA2022/DATA-P/Camera/CIMG7864.JPG
# GoPro
# https://nspracticum.science.uu.nl/DATA2022/DATA-P/Camera/G0010020.JPG

directory = 'C:/Temp/Camera-Voorbereiding/'
photo = 'CIMG7864.JPG'

# Import picture (single frame) in Python
frame0 = cv2.imread(directory + photo)
# A photo usually has high resolution (4608x3456 for Exilim example,
# 4000x3000 for GoPro example)
# We decrease this to 640x480, which is typical for video
frame = cv2.resize(frame0,(640,480))
### Display
# When using OpenCV imshow, you have to give your frame a name
cv2.imshow('fig1',frame)
# The combination waitKey + destroyWindow can display the frame for some time
# after deleting it; waitKey(xx) displays the image for xx ms
#cv2.waitKey(10000)
# Close the figure using destroyWindow
#cv2.destroyWindow('fig1')
# Write the resized frame to file
cv2.imwrite(directory + 'test1.png',frame)
```

- Sluit de terminal (typ `exit` and sluit het venster)
- Open Anaconda en verander 'Applications on' naar openCV trackpy. Start dan Spyder (in de nieuwe environment opencv) en alles zou moeten werken.

➡ Vervang steeds de directory door een voor jou relevante locatie. We gaan ervan uit dat je figuren niet inline maar in aparte vensters weergeeft.

1.3 Coördinaten en schaling

Wanneer je een frame inleest in Python, dan gebeurt dat in de vorm van een array. Elk element in die array correspondeert met de waarden van een zekere pixel (zie ook Sectie 1.4). Voor de oriëntatie is het goed om iets te zeggen over welke pixel correspondeert met welke positie in het array (zie ook Fig. 1):

- Als je over het figuur beweegt met je muis, dan zie je dat de oorsprong van het figuur linksboven ligt. In OpenCV is de x -richting de horizontale richting (naar rechts), en de y -richting de verticale richting (naar beneden). De pixel linksonder in het naar 640×480 herschaalde frame heeft bijvoorbeeld $x = 0$, $y = 479$.
- In het array ligt de oorsprong eveneens linksboven. De rij is de verticale richting (naar beneden), de kolom de horizontale richting (naar rechts). Als je de pixel linksonder wilt selecteren, dan moet je dus invoeren `frame[479, 0]`.

Oftewel, de coördinaat x correspondeert met de *kolom*-positie (tweede index) in de array, en de coördinaat y met de *rij*-positie (tweede index). Dat is niet helemaal intuïtief.

In frames is de ‘natuurlijke’ schaal de pixel. Bij het volgen van objecten zul je daarom altijd de positie uitgedrukt krijgen in pixel. Voor interpretatie is echter de werkelijke afstand vereist. In een frame moet daarom ergens een maat aanwezig zijn waarmee de schalingsfactor voor het omzetten van de afstanden in pixel naar fysieke afstanden bepaald kan worden. In Fig. 1 is dit bijvoorbeeld een liniaal, in Opdracht 1 zullen we het object zelf gebruiken om de schaalfactor te bepalen.

Een nauwkeurige bepaling kun je doen met behulp van Fiji². Met de knop met het lijntje kun je de lengte van een lijn in pixels bepalen.

➡ Let op dat de schalingsfactor afhangt van de gekozen geometrie. Als je bijvoorbeeld de camera op een andere hoogte boven de opstelling hangt, zal de factor veranderen.

Opdracht 1

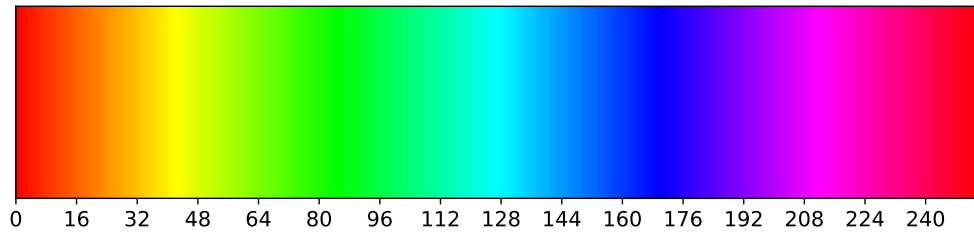
Opname maken

- Maak met het statiefmateriaal een standaard waar de camera in opgehangen kan worden. Richt de camera naar beneden, op de labtafel.
- Leg twee gekleurde pingpongballen en een liniaal neer en maak een foto (doe dit in eerste instantie zonder flits).
- Kopieer je foto/frame via de bijgeleverde kabel met USB-aansluiting naar de zaalcomputer of je laptop om verder te verwerken.

OpenCV

- Installeer OpenCV volgens de procedure in Sectie 1.2. Installeer naar keuze Fiji op je laptop.

²Fiji (Fiji is just ImageJ) is een beeldbewerkingsprogramma dat in de biomedische wetenschappen veel wordt gebruikt. Op de practicum-pc's is het standaard geïnstalleerd; zoek op Fiji. Op <https://imagej.net/Fiji/Downloads> kun je het programma gratis downloaden om op je laptop te installeren. Unzip en zet de directory op een geschikte plek (bijvoorbeeld in Programs).



Figuur 2: HSV-kleurwaardes op een schaal 0 tot 255.

- Open Anaconda en/of Spyder. Start een nieuw script en importeer de packages `numpy` en `matplotlib.pyplot` zoals je gewend bent.
- Importeer OpenCV volgens `import cv2`.
- Importeer, herschaal, plot en schrijf je frame met behulp van de voorbeeldcode in Sectie 1.2. Evalueer het script en kijk of er geen foutmeldingen gegenereerd worden.

Schaling

- Bepaal de schalingsfactor voor het frame dat je zojuist hebt gemaakt. Controleer of de factor correct is door de diameter van een pingpongbal in pixel te meten, deze om te rekenen naar een aantal cm en de uitkomst te vergelijken met een direct gemeten diameter.
- De schalingsfactor zal in de praktijk natuurlijk een onzekerheid kennen. Hoe werkt deze onzekerheid door in de analyse van de metingen? En hoe werkt een verkeerd bepaalde schalingsfactor door? Is het zinvol de bepaling van de schalingsfactor te reproduceren (en zo ja, op welke manier)?

Demonstreer kort je werk aan de assistent.

Opdracht 1 gaat op p. 9 verder.

1.4 Kleurenruimte en masking

Het in Python gecreëerde frame is een drie-dimensionaal (numpy) array. De eerste twee dimensies zijn de grootte van het figuur in pixel (bijvoorbeeld 640×480). Elke pixel wordt daarnaast beschreven door een *kleurcode* van drie getallen³. De standaard codering van kleur bij importeren in Python is BGR (Blauw-Groen-Rood). Elke kleur kent 8-bit oftewel $2^8 = 256$ intensiteitswaarden. Bijvoorbeeld rood is `[0,0,255]`.

De BGR-codering is echter niet optimaal wanneer je op basis van kleuren onderscheid wilt maken. Een kleur als geel kan bijvoorbeeld op veel manieren gerealiseerd worden op de BGR-schaal. Vandaar dat in tracking vaak gebruik wordt gemaakt van een andere codering, namelijk HSV (Hue-Saturation-Value), ook wel HSB (Brightness) genoemd. De Hue definieert uniek de kleur; Saturation en Value geven de 'grijsheid' en de sterkte aan. In Fig. 2 zie je de Hue-kleurwaarden weergegeven op een schaal 0 - 255. Met de Hue is het eenvoudiger een object te selecteren. We converteren eerst van BGR naar HSV kleurenruimte.

```
# Change colorspace
frame_hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
```

³Voor meer informatie over kleurenruimtes: https://en.wikipedia.org/wiki/Color_space

Merk op dat de informatie in het figuur niet veranderd is, die informatie wordt alleen uitgedrukt in drie andere 'grootheden' met andere waarden.

Om een object te identificeren, maken we gebruik van een masker (*mask*) op basis van de kleur en in mindere mate de intensiteit van het object. Zo'n masker geeft een waarde 1 als het aan de selectievoorwaarden voldoet, en 0 als dat niet zo is. Hiervoor moeten we eerst weten wat de HSV-waarden zijn van het object.

Een goede aanpak is om je eerder gemaakte eerste frame in te laden in Fiji en via Image | Type | HSB Stack het te converteren naar drie plaatjes die resp. H, S en V weergeven. De waarde *value* = in de menubalk geeft de waarde ter hoogte van de cursor. Je kunt hier goed zien of een opname overbelicht (geen goede H-waardes, heel hoge V-waardes), onderbelicht (mogelijk goede H-waardes maar heel lage V-waardes) of goed belicht is.

Laten we proberen de rozerode bal in Figuur 1 te volgen; deze kleur komt weinig voor in de achtergrond. In de HSV-kleurenkaart Fig. 2 zien we dat roze overeenkomt met Hue ~ 220 , maar een blik in Fiji geeft aan dat een waarde rond de 6 beter is⁴.

In Code 2⁵ definiëren we de onder- en bovengrens van de Hue en de Value waarbinnen we gegevens 'doorlaten'; de rest van het figuur wordt op nul gezet⁶.

Op basis van proberen kun je erachter komen welke grenzen goed werken. We kiezen voor de roze bal een bandbreedte in de Hue van 10, en in eerste instantie de ondergrens voor Saturation en Value op respectievelijk 30 en 100 (en beide bovengrenzen op de maximale waarde van 255).

Code 2: Definitie en toepassing masker.

```
# NOTE: this segment of code is not self-contained,
# but must be incorporated in own code before compilation

hue0 = 6
delta_hue = 10
# Scaling factor because Hue in Python scales from 0 to 179 (0 to 255 in ImageJ)
hue_fac = 180/255
# Lower and upper limits for H, S and V in the mask
hmin = hue_fac*(hue0-delta_hue)
hmax = hue_fac*(hue0+delta_hue)
smin = 30
smax = 255
vmin = 100
vmax = 255
lower = np.array([np.maximum(0,hmin),smin,vmin])
upper = np.array([np.minimum(180,hmax),smax,vmax])
# Mask for the frame, and the original frame removing 'non-object'
# Note that Hue can also 'loop around zero'
if hmin<0:
    mask=cv2.inRange(frame_hsv,lower,upper)+\
        cv2.inRange(frame_hsv,np.array([np.mod(hmin,180.),smin,vmin]),\
            np.array([180.,smax,vmax]))
elif hmax>180:
    mask=cv2.inRange(frame_hsv,lower,upper)+\
        cv2.inRange(frame_hsv,np.array([hmax-180.,smin,vmin]),\
            np.array([180.,smax,vmax]))
```

⁴Hoewel de bal er op het oog roze uitziet, is de werkelijke kleur dus eerder rood. Het 'roze-achtige' is blijkbaar omdat de bal tegen overbelichting aan zit.

⁵Let op: vanaf nu is de code niet compleet, maar steeds een stuk voorbeeldcode dat je in je eigen script moet integreren.

⁶In deze code hebben we er voor je rekening mee gehouden dat de Hue in Python loopt van 0 t/m 179 in plaats van 0 t/m 255. Verder sluit de Hue-schaal bij 255 aan op 0, wat in principe voor problemen kan zorgen bij waarden die dichtbij 255 / 0 liggen. We hebben daarom ook met `np.mod` een 'rondlopend' interval gemaakt.



Figuur 3: Foto Fig. 1 met masker uit voorbeeldcode toegepast. Het object is duidelijk geselecteerd, maar ook delen van de achtergrond zijn nog zichtbaar.

```

cv2.inRange(frame_hsv,np.array([0.,smin,vmin]),\
            np.array([np.mod(hmax,180.),smax,vmax]))
else:
    mask=cv2.inRange(frame_hsv,lower,upper)
res=cv2.bitwise_and(frame,frame,mask=mask)
# Plot of masked object
cv2.imshow('mask', mask)
cv2.imwrite(directory + 'test-mask.png',mask)
cv2.waitKey(1000)
cv2.destroyWindow('mask')
# Plot of masked object in original color
cv2.imshow('res',res)
cv2.waitKey(1000)
cv2.destroyWindow('res')

```

Fig. 3 laat het resultaat van het masker zien bij bovenstaande parameters, met het object in zijn oorspronkelijke kleur. De vraag is of er misschien op basis van de H-, S- en V-waardes betere grenzen te definiëren zijn dan in dit voorbeeld (we hebben nu zeer ruime marges gekozen).

➡ Om zelf te proberen: zet bijvoorbeeld de ondergrenzen voor S en V hoger totdat de achtergrond verdwenen is. Deze foto blijkt achteraf tegen overbelichting aan te zitten; ook delen van de achtergrond scoren hoog op de parameter V.

1.5 Aandachtspunten voor volgbaarheid object

In een tracking-algoritme onderscheiden we het object vaak naar *kleur* en *intensiteit*. Om een object te kunnen volgen, moet het op de één of andere manier “afsteken” tegen de omgeving. De afgelopen jaren is vaak gebleken dat studenten problemen ondervonden in de *tracking*-fase van het experiment. Dit is vaak te herleiden naar verkeerde keuzes in ontwerp van de meting (kleurkeuzes, belichting, *framerate*). Dit voorwerk is dus cruciaal! Belangrijke aanwijzingen:

- Het belangrijkste: **Zorg ervoor dat de kleur van het object niet in de achtergrond voorkomt!** Dit kun je doen door tegen een egale achtergrond (wit, maar liever zwart) te werken, en door een object

te gebruiken met daarmee sterk contrasterende kleur of markering. In de projectomschrijving beschrijven we methodes om een statische achtergrond te elimineren.

- Het voordeel van een zwarte of donkere achtergrond boven een lichte is dat het object zowel onderscheidbaar is naar kleur als naar intensiteit.
- Er zijn gekleurde stipjes beschikbaar die je als *markers* kunt gebruiken voor vlakken op het te volgen object als dat niet te ver van de camera is. Voor objecten op grotere afstand zul je zelf grotere stippen moeten maken, of het object zelf kleuren. Geef het bij de practicumleiding aan als je hiervoor verf, stift, plakband o.i.d. nodig hebt. Wees creatief, het gaat erom dat de tracking werkt.
- Houd er rekening mee dat je object in een video naar een andere locatie beweegt. Dit heeft meestal geen sterke gevolgen voor de Hue, maar kan wel effect hebben op S en V. Bijvoorbeeld omdat het object uit een goed belichte zone beweegt. Het is daarom aan te raden om de grenzen van het masker voor deze twee parameters (in eerste instantie) niet te strak te zetten.

1.6 Positiebepaling

Je hebt een masker gemaakt, en nu is de vraag hoe dit masker zo te interpreteren dat ‘dè’ positie van het object wordt gevonden. Als ‘dè’ positie van het object gebruiken we het *zwaartepunt*. In het masker hebben (bijna) alle objectposities sterkte 1, en (bijna) alles daarbuiten sterkte 0. Je zou dus zeggen, neem de gemiddelde positie van alle pixels met sterkte ongelijk aan nul en je hebt je positie. Helaas kunnen er ook toevallig pixels op 1 staan ver buiten het eigenlijke object en die zouden zo’n berekening ernstig verstoren.

Een tussenstap verloopt via het gebruik van *contouren*⁷. Kort gezegd bepaal je alle contouren (‘omtrekken’) van de objecten in het frame na masking, selecteer je het object met de grootste contour, en bepaal je van dat contour het zwaartepunt.

Code 3: Bepalen contouren en zwaartepunt van grootste object.

```
# NOTE: this segment of code is not self-contained,
# but must be incorporated in own code before compilation

# Find all contours present in the mask
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                        cv2.CHAIN_APPROX_SIMPLE)[-2]

# Test for presence of contours
if len(cnts)==0:
    # If not present (object has left image plane or is otherwise not found),
    # set position to [-1,-1]
    pos=[-1,-1]
else:
    # Otherwise pick the contour with largest enclosed area and
    # determine 'center of mass' (CM)
    c = max(cnts, key=cv2.contourArea)
    M = cv2.moments(c)
    center = ((M["m10"] / M["m00"]), M["m01"] / M["m00"])
    pos=np.array([center])

# Diagnostics: draw contour (yellow) and CM (white) of object
frame2 = frame
cv2.drawContours(frame2, c, -1, (0,255,255), 3)
cv2.circle(frame2, (np.int_(center[0]),np.int_(center[1])), 5, (255, 255, 255), -1)
cv2.imshow('fig2',frame2)
```

⁷https://docs.opencv.org/4.5.4/d4/d73/tutorial_py_contours_begin.html

Opdracht 1 (vervolg)

► Het is niet verplicht, maar wel zeer aanbevolen om delen van de code in de vorm van functies (**def**) te gieten. Dat maakt dat je in de toekomst een analyse voor een nieuwe situatie sneller kunt opzetten.

Masking

- Kopieer Code 2 in je Python-script.
- Speel wat met de HSV-waarden die we default hebben gekozen voor de roze bal. Wat zijn correcte grenzen waarbinnen uitsluitend de bal gevonden wordt? Wat gebeurt er wanneer het Hue-bereik wordt opgerekt tot ± 30 ?
- Maak zelf een geschikt masker voor de blauwe bal en demonstreer de werking ervan.

Positiebepaling

- Gebruik Code 3 om de positie van de rode bal (in pixel) te bepalen.

Demonstreer kort je werk aan de assistent.

Rapporteer na afronding in je labjournaal. Bepaal zelf wat vereist is voor een volledige rapportage (gebruik eventueel de rubric). Zaken die je kunt opnemen zijn geproduceerde code, figuren voor en na masking, en een aantal observaties omtrent kleuren en selectie. Waarom is wit bijvoorbeeld geen geschikte keuze voor de kleur van een object? Zijn de kleuren van de ballen goed gekozen in relatie tot de achtergrond?

► Dit is het einde van Opdracht 1. Als je dit punt voor het einde van de contacturen hebt bereikt, is het aan te raden alvast door te werken in Opdracht 2, die wat groter is.

2 Tracking

In deze tweede opdracht bouwen we het script uit de eerste opdracht uit tot een volwaardig tracking-script. Daarna pas je je script toe op een botsing van twee pingpongballen.

2.1 Openen en uitlezen van een video-kanaal

Op Blackboard kun je korte filmpjes vinden van een slinger (slinger.MOV, slinger.MP4), opgenomen met respectievelijk de Exilim-camera en de GoPro-camera, om de tracking uit te proberen. Fig. 4 toont het eerste frame van zo'n opname.

Code 4 laat zien hoe je een videokanaal opent⁸, en alle frames één voor één leest. We vragen ook wat handige meta-informatie op, namelijk de framerate en het totaal aantal frames⁹

Code 4: Openen videokanaal en lezen frames.

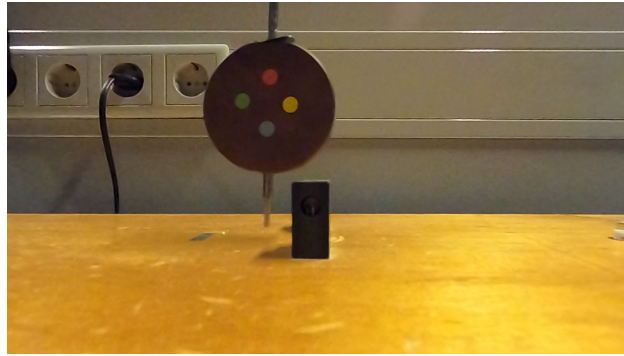
```
# location video file (Exilim)
# You can also download the example videos to a local directory
dir = 'http://nspracticum.science.uu.nl/DATA2022/DATA-P/Camera/'
# Exilim video
video='slinger.MOV'
# GoPro video
# video='slinger.MP4'
# Define video capture channel
cap=cv2.VideoCapture(dir+video)

# Meta-information
# Total number of frames
Nframes = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
# Frames per second (fps) - incorrect for Exilim, set manually
#fps = cap.get(cv2.CAP_PROP_FPS)
fps = 240

# Loop through frames
for nframe in np.arange(Nframes):
    print("Frame " + str(nframe))
    # cap.read() reads the next frame (image) of the video
    # found is a boolean, True if the frame is found
    # frame contains the frame (image) data
    found, frame = cap.read()
    # This is a safeguard for the GoPro camera, which sometimes contains
    # a defect frame. This way, we skip it if found is False, and proceed
    while not found and nframe < Nframes-1:
        print('error reading frame ' + str(nframe) + ', skipping')
        nframe += 1
        cap.set(cv2.CAP_PROP_POS_FRAMES, nframe)
    found, frame=cap.read()
```


⁸Dit kan een bestaande opname zijn, maar je kunt met `cv2.VideoCapture(0)` bijvoorbeeld ook de opnames van een verbonden webcam inladen.

⁹NB de Exilim geeft altijd 30 fps voor de framerate. Waarom is ons niet bekend. Als dat zo is, gebruik dan de waarde die je zelf op de camera hebt ingesteld, 240 fps in het voorbeeld.



Figuur 4: Eerste frame uit voorbeeldvideo van een slinger.

Opdracht 2

► Het is aan te raden om bij het ontwikkelen van de code de regels stap voor stap te laden middels selectie en Ctrl + Enter, of via icoon  in de menubalk. Met `###` kun je je script in cellen delen. Daarna kun je met Ctrl + Enter de code cel voor cel evalueren.

► Maak en bekijk bij het ontwikkelen van de code veel tussentijdse plots; die kunnen je vaak snel vertellen hoe je je HSV-parameters moet kiezen, of je framerate hoog genoeg is, etc. Als alles eenmaal werkt, kun je plotopdrachten uitcommenten met `#` of naar keuze activeren middels `if`-statements.

Tracking

- Start een nieuw script, laad de benodigde packages, en open met Code 4 een voorbeeldfilmpje.
- Schrijf het eerste frame van het voorbeeldfilmpje weg om dit te kunnen bestuderen met Fiji¹⁰.
- Gebruik je code uit de eerste opdracht om de `while`-loop te vullen. Kies een stip op de slinger om te volgen, bepaal goede HSV-waarden voor het masker en bekijk of het masker werkt. Merk op dat we vier kleuren stippen hebben opgeplakt (rood, geel, groen, blauw), waarvan de één beter afsteekt tegen de achtergrond dan de andere. Maak zelf een overweging omtrent welke stip je wilt volgen (of misschien wil je er wel meerdere tegelijk volgen?).
- Breid je code uit tot een routine die na het eerste frame steeds een volgend frame i inleest, de objectpositie(s) $[x_i, y_i]$ bepaalt en deze wegschrijft in een `numpy`-array `pos`. Als er geen object wordt gedetecteerd (omdat het buiten beeld verdwijnt of niet meer aan de eerder gestelde voorwaarden aan kleur of intensiteit voldoet), wordt er `[-1, -1]` weggeschreven.
- Bedenk hoe je een `numpy` array maakt met alle tijden t_i .
- Maak een plot van $x(t)$ en $y(t)$ en controleer dat er inderdaad sprake is van slingering. Wat is voor deze tracking de onzekerheid in de bepaalde positie tracking ongeveer (in pixel)?

Demonstreer tot slot de werking van je code aan de assistent.

Opdracht 2 gaat verder op p. 13. Eerst geven we wat aanwijzingen voor de instelling van de camera's en het tracken van objecten.

¹⁰De schaling is in dit geval nog niet zo belangrijk, maar als je de schalingsfactor ook wilt instellen: de diameter van de slingerschijf is ongeveer 7 cm

2.2 Camera-instellingen

► Controleer steeds je camera-instellingen. Als de camera uit is geweest of door iemand anders gebruikt, kunnen de instellingen gewijzigd zijn!

► Zorg ervoor dat de bestanden niet te groot worden. Een bestandsgrootte die zonder problemen verwerkt kan worden is een paar honderd MB. Als indicatie: bij 240 fps correspondeert 100 MB voor beide cameratypes met een filmpje van ± 15 seconde.

Exilim EX-ZR10 De Exilim EX-ZR310 wordt als volgt ingesteld:

1. Zet de camera in de opnamemodus (knop met rode fototoestel) met de draaiknop op Automatisch (het rode rechthoekje)
2. Ga via de knop Menu (rechtsonder) naar het tabblad Kwaliteit
3. Ga met de pijltjestoetsen (grote zwarte knop) naar (icoon Camera) Kwaliteit.
4. Door hier naar rechts te klikken krijg je een lijst waaruit je een opnamesnelheid kunt kiezen. Als je even op een item blijft staan, zie je de instellingen van de camera bij die keuze. HS120 of HS240 zijn aanbevolen. Informatie over de opnamesnelheid is te vinden in de handleiding (zie Blackboard), pagina 70-71.
5. Druk op SET (midden zwarte knop) om de instelling te bewaren.

GoPro Hero 9 In tegenstelling tot de Exilim heeft de GoPro een touchscreen. Links op de camera zit de aan/uitknop. Onder het paneeltje rechts zit de USB-connectie en de oplaad-connectie.

1. Door links en rechts te swipen kun je kiezen voor Photo of Video modus (of Time Lapse - voor als je een zeer traag experiment wilt doen)
2. Met de centrale knop onderin kun je de Video Settings wijzigen. Standaard staat dit op 1080p, 60 fps, Wide opname.
3. Met het potloodje kun je de instellingen wijzigen. In ieder geval moet je de Wide lens naar een (digitale) lens Linear of Narrow (smaller beeld) zetten, anders wordt het beeld vervormd (kijk bij Wide bijvoorbeeld maar eens naar rechte lijnen in de zaal). Met de rode knop kun je opnames starten en stoppen. Swipe naar boven om opnames terug te kijken.

2.3 Object tracken als functie van de tijd

Als het object in het eerste frame goed te zien is, en scherpte, belichting et cetera niet veel veranderen als functie van de tijd, dan zou het object goed te volgen moeten zijn. Vaak gaat het echter fout omdat de belichtingssterkte verandert, of omdat het object heel snel gaat bewegen.

Oplossingen zitten soms in het verbeteren van het experiment, soms in aanpassingen in de tracking-code, en soms in beide. Enkele typische problemen en suggesties:

- **Probleem:** object is bijna direct na eerste frame niet goed meer te identificeren.
Oorzaak: in het eerste frame is te weinig (alleen het centrum van een stip) of juist te veel (stip inclusief achtergrond) geselecteerd.
Oplossingen: probeer de zichtbaarheid van het object in of het masker voor het eerste frame te verbeteren.

- **Probleem:** tracking-routine vindt punten in de achtergrond i.p.v. het gewenste object.
Oorzaak: de achtergrond bevat blijkbaar ook de kleur van het object.
Oplossingen: geef het object een andere kleur, gebruik extra masks (zie instructie projectfase), beperk de aanwezigheid van de kleur in de achtergrond. Verbeteren van de belichting kan ook helpen.
- **Probleem:** object wordt vaag bij hoge snelheden.
Oorzaak: de sluitertijd is te lang.
Oplossingen: selecteer een hogere framerate (b.v. twee keer zo hoog; sla in de verwerking eventueel 1 op 2 frames over om de grotere hoeveelheid frames te kunnen verwerken).
- **Probleem:** object raakt verloren na verloop van tijd.
Oorzaken: te grote snelheid(sverschillen); verandering in belichtingssterkte; overlap met achtergrond.
Oplossingen: verbeter belichting, elimineer de effecten in de achtergrond, probeer een hogere framerate (om de afstand tussen de objecten in opeenvolgende frames te verkleinen).

Het vraagt enige oefening om tracking goed te krijgen. Vraag in een vroeg stadium de assistent of docent om hulp als het echt niet lukt om iets zinnigs met je opnames te doen.

2.4 Keuze voor framerate

Omdat je bewegende objecten gaat volgen, moet je in de verkenningsfase nadenken over de snelheid van het proces dat je wilt volgen en derhalve de opnamesnelheid.

- De standaardinstelling van de Exilim is 30 fps, van de GoPro 60 fps. Wij adviseren bij wat snellere processen een opnamesnelheid van minstens 120 of 240 fps, omdat bij de standaard 30 frames per seconde de sluitertijd zo lang is dat je dan onscherpe beelden krijgt. Elk frame bevat voor de Exilim dan 640×480 (120 fps) of 512×384 (240 fps) pixels, voor de GoPro dan 2704×1520 (120 fps) of 1920×1080 pixels (240 fps).
- Bij opnames met hoge framerate moet je extra aandacht schenken aan belichting (de hoeveelheid licht die per frame invalt wordt lager)! Gebruik dan de bureaulampjes of bij grotere experimenten de bouwlampen.
- Er kan met de Exilim ook bij hogere fps gemeten worden (480 of 1000 fps), maar dan verlies je naast lichtsterkte ook resolutie.
- Probeer van tevoren te bedenken wat een goede effectieve framerate zou zijn om je meting te kunnen analyseren. Bijvoorbeeld door de maximale snelheden van objecten op basis van theorie of een proefopname af te schatten, of door omstandigheden (massa's, valafstanden etc.) zo te kiezen dat de snelheid geen drempelwaarde overschrijdt.

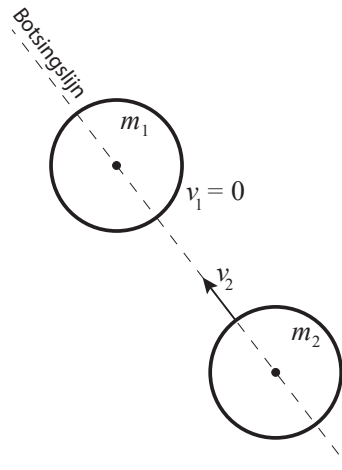
Opdracht 2 (vervolg)

We beschouwen de botsing van twee pingpongballen, noem ze 1 en 2. Behoud van kinetische energie¹¹ en impuls in de meest algemene vorm kan men geven in vectorvorm:

$$(m_1 \vec{v}_1)_{\text{voor}} + (m_2 \vec{v}_2)_{\text{voor}} = (m_1 \vec{v}_1)_{\text{na}} + (m_2 \vec{v}_2)_{\text{na}} \quad (1)$$

$$\left(\frac{1}{2} m_1 |v_1|^2\right)_{\text{voor}} + \left(\frac{1}{2} m_2 |v_2|^2\right)_{\text{voor}} = \left(\frac{1}{2} m_1 |v_1|^2\right)_{\text{na}} + \left(\frac{1}{2} m_2 |v_2|^2\right)_{\text{na}} \quad (2)$$

¹¹Uiteraard heeft een rollende bal ook rotatie-energie. De rotatie-energie van een object wordt gegeven door $\frac{1}{2} I \omega^2$, met I het traagheidsmoment en ω de hoeksnelheid. Voor een bol geldt dat $v = \omega r$. Men kan laten zien dat I voor een (holle) bol zoals een pingpongbal evenredig is met v^2 , net als de kinetische energie. Oftewel, de rotatie-energie is steeds een fractie van de kinetische energie. Vanwege deze reden kunnen we de rotatie-energie buiten beschouwing laten.



Figuur 5: De in Opdracht 2 te bestuderen botsing van pingpongbal 2 met een stilliggende pingpongbal 1.

We gaan er nu vanuit dat de snelheden $\vec{v}_1 = v_1$ en $\vec{v}_2 = v_2$ langs de botsingslijn liggen (we hoeven dan het vectoriële karakter van de snelheden niet te betrekken). Verder beschouwen we de eenvoudige situatie dat één bal stilligt en de andere er tegenaan botst¹², zoals getoond in Fig. 5. De behoudswetten vereenvoudigen dan tot

$$(m_2 v_2)_{\text{voor}} = (m_1 v_1)_{\text{na}} + (m_2 v_2)_{\text{na}} \quad (3)$$

$$\left(\frac{1}{2} m_2 v_2^2\right)_{\text{voor}} = \left(\frac{1}{2} m_1 v_1^2\right)_{\text{na}} + \left(\frac{1}{2} m_2 v_2^2\right)_{\text{na}} \quad (4)$$

Opdrachten

- Wanneer bal 1 stilligt en bal 2 met een snelheid $(v_2)_{\text{voor}}$ komt aanrollen, leid dan de uitdrukkingen voor de snelheden $(v_1)_{\text{na}}$ en $(v_2)_{\text{na}}$ na de botsing af.
- Voor het wegen van de pingpongballen kun je de nauwkeurige balansen in MIN0.01 gebruiken. Mogelijk kun je gegeven de meetnauwkeurigheid gebruik maken van $m_1 = m_2 = m$.
- Hoe moet je het experiment opzetten en welke gegevens heb je nodig om te controleren of er sprake is van behoud van energie en impuls? Enkele tips om je op gang te helpen:
 - Beschrijf hoe je snelheden kunt bepalen uit de gemeten posities.
 - Het is natuurlijk toegestaan om het experiment in de verkenningsfase enkele malen te herhalen totdat je een geschikte opzet hebt gevonden (belichting, framerate, zorgen voor een correcte botsing). Uiteraard rapporteer je kort in je labjournaal hoe je tot de ‘ideale’ opzet bent gekomen.
 - Neem de overwegingen van voorgaande opdrachten mee in je experimentontwerp.
 - Er hoeft slechts één botsing geanalyseerd te worden (al mag je er meerdere analyseren als de tijd dat toelaat) en een bespreking van onzekerheden mag kwalitatief plaatsvinden.
- Voer het experiment uit, rapporteer totale kinetische energie en impuls voor en na de botsing, en bediscussieer je bevindingen.

Rapporteer na afronding in je labjournaal. Bepaal zelf wat vereist is voor een volledige rapportage (gebruik de rubric en de tussentijdse opdrachten).

¹²In de projectfase kun je overwegen om willekeurige botsingsprocessen te beschrijven. Een nuttige link hierbij: <http://www.vobarian.com/collisions/2dcollisions2.pdf>
