Training 4.0: OpenCV

Search the course Search Start Course

This course is ending in 1 week on Jun 1, 2022.

After this date, course content will be archived.

Collapse All

- Johdanto OpenCV:hen ja kurssin esittely
  - Video: Johdanto OpenCV:hen
  - Johdanto ja kurssin rakenne
- Asennus, käyttöönotto ja dokumentaatio
  - Video: asennus ja käyttöönotto
  - Asennuslinkit ja lisätietoa
  - Video: OpenCV:n dokumentaatiosivujen lukeminen
- Harmaasävykuvat ja objektien erottelu
  - Video: objektien tunnistus koon perusteella
  - Videoesimerkin koodi ja lisätietoa
  - Harjoitus
- Värikuvat ja värien esittäminen
  - Video: värikuvat ja värien tunnistaminen
  - Videoesimerkin koodi ja lisätietoa
  - Harjoitus
- Monimutkaisemmat kuvaoperaatiot
  - Monimutkaisemmat kuvaoperaatiot: johdanto
  - Video: monimutkaisemmat kuvaoperaatiot
  - Videoesimerkin ympyrähaun koodi ja lisätietoa
  - Harjoitus: ympyrähaku
  - Videoesimerkin piirrehaun koodi ja lisätietoa
  - Harjoitus
- Tekstintunnistus, viivakoodit ja QR-koodit
  - Video: tekstin ja viivakoodien tunnistaminen ja lukeminen
  - Videoesimerkin tekstin lukemisen koodi ja lisätietoa
  - Videoesimerkin viivakoodin lukemisen koodi ja lisätietoa

Upcoming Dates

Jun 1, 2022

Course End
After this date, course content will be archived.

Harjoitus

Kuvan lukeminen kameralta

Video: kuvan lukeminen web-kameralta

Kuvan lukeminen kameralta

Video: konenäkökameran käyttö

Harjoitus

Ekstra: virtuaaliympäristöt

Video: virtuaaliympäristön rakentaminen ja käyttäminen

Johdantovideo

Bookmark this page

Johdanto OpenCV:hen

STAFF DEBUG INFO

Kurssin esittely

Bookmark this page

OpenCV on erittäin kattava avoimen lähdekoodin konenäkökirjasto. Se sai alkunsa Intelin projektista 1999, ja sitä pidetään edelleen aktiivisesti yllä. OpenCV on laajasti käytetty yritysmaailmassa (myös suurissa yrityksissä kuten Intel ja Google), korkeakouluissa ja julkisyhteisöissä. Opencv.orgin mukaan sen käyttäjäyhteisöön kuuluu yli 47 tuhatta ihmistä ympäri maailman ja kirjastoa on ladattu historiansa aikana arviolta yli 18 miljoonaa kertaa.

Opetuksellisesti OpenCV:n hyvä puoli on sen avoimuus. Kaikkien kuvaoperaatioiden tulokset saa näkyville ja jokaisen parametrin vaikutuksen voi kokeilla. Jos kaikki likaisimmat yksityiskohdat kiinnostavat, voi itse lähdekoodia käydä lukemassa. Koska kaupalliset kirjastot ja tuotteet kuten älykamerat käyttävät samoja menetelmiä, menetelmien syvällinen ymmärtäminen auttaa myös kaupallisten tuotteiden käytössä.

Tämä kurssi pyrkii antamaan kävijälleen katsauksen OpenCV:n maailmaan ja näyttämään muutamia erilaisia esimerkkejä siitä, mihin sen eri toimenpiteitä voi käyttää. Kurssin tarkoitus ei ole olla perustellinen OpenCV-opas, vaan herättää opiskelijan mielenkiinto alkaa kokeilla konenäköä ja OpenCV:tä omiin tarkoituksiinsa ja etsiä lisää tietoa netistä. Kurssilla käydään läpi seuraavat asiat:

1. Pythonin, OpenCV:n ja Visual Studio Code -kehitysympäristön asentaminen ja käyttöönotto sekä OpenCV:n dokumentaation lukeminen
2. Objektien lajittelu niiden ääriviivojen perusteella
3. Värien esitys kuvassa ja värintunnistus
4. Monimutkaisemmat operaatiot: ympyröiden etsintä, piirteet ja kulmat
5. Tekstin, viivakoodien ja QR-koodin tunnistus
6. Kuvan lukeminen kameralta
7. Ekstra: virtuaaliympäristön rakentaminen ja käyttö

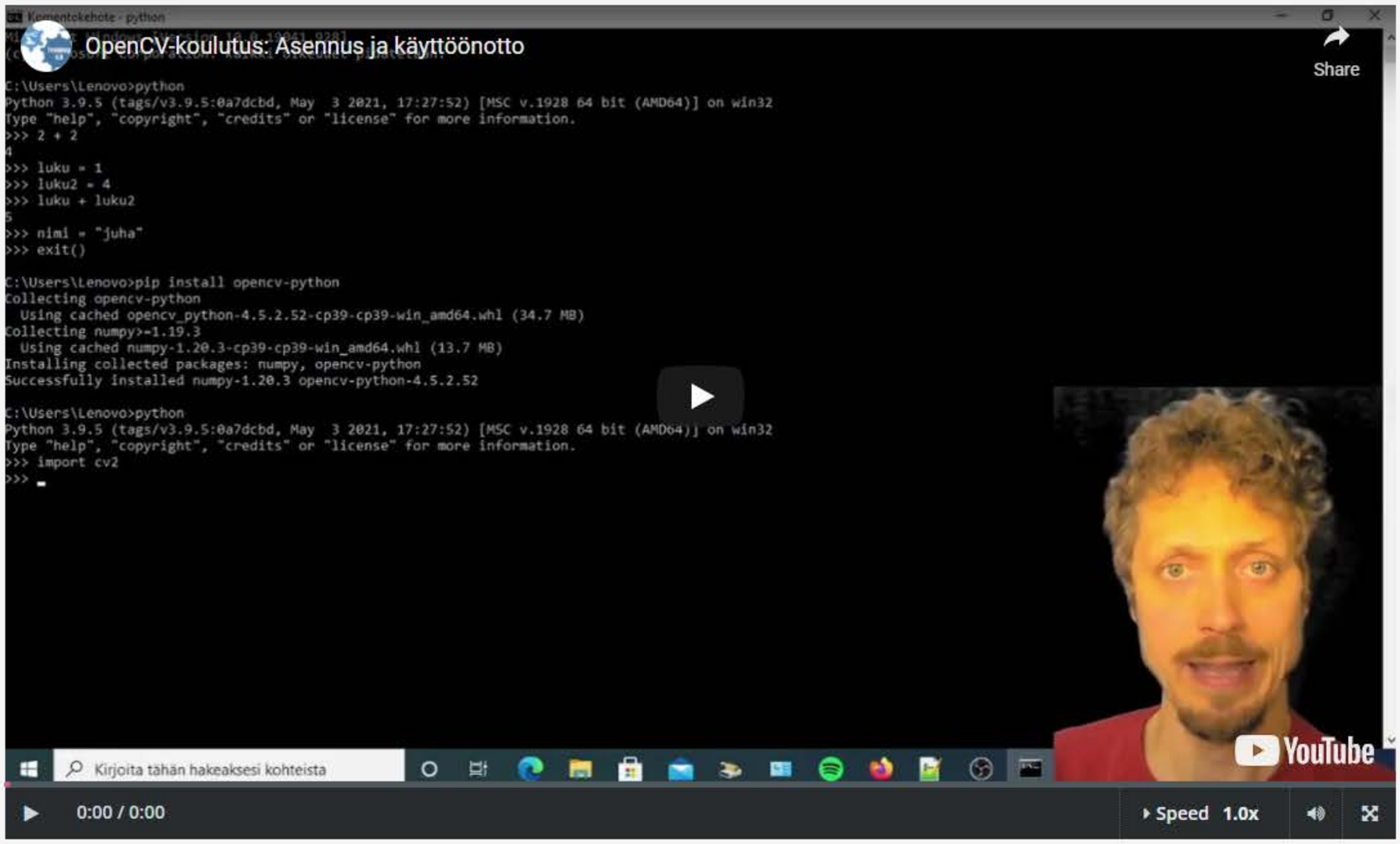
Jokainen osio koostuu videosta, joissa kyseinen osa-alue esitellään koodiesimerkin kautta, koodiesimerkistä kommentteineen sekä harjoituksesta, jossa kyseistä koodiesimerkkiä pitää muokata ja löytää sen perusteella vastaukset kysymyksiin. Tällä lailla kurssin kävijä tulee tutustuneeksi osa-alueeseen hieman laajemmin kuin videolla käydään läpi.

STAFF DEBUG INFO

Asennusvideo

Bookmark this page

Asennus ja käyttöönotto



STAFF DEBUG INFO

Linkit ja lisätiedot

Bookmark this page

Python on korkean abstraktiotason tulkkipohjainen ohjelmointikieli. Korkea abstraktiotaso tarkoittaa sitä, ettei koodaajan tarvitse huolehtia kaikista teknisistä yksityiskohdista, vaan komennot annetaan hyvin ylätasolla. Vähän kuin ajaaksesi autoa sinun ei tarvitse tietää, kuinka auto toimii. Tulkkipohjaisuus tarkoittaa puolestaan sitä, että ohjelmakoodia ei käännetä konekieliseksi ajettavaksi tiedostoksi (esim. exeksi) ennen ohjelman suorittamista, vaan koodia suoritetaan ikään kuin lennosta suoraan Pythonilla kirjoitetusta tiedostosta (tiedostopääte py). Tämä tarkoittaa myös sitä, että Python-koodia ei pysty suorittamaan ilman koneelle asennettua Python-tulkkia (toisin kuin konekielisiä exe-tiedostoja). Python on verrattain helppo oppia ja sille on olemassa valtava määrä erilaisia ilmaisia kirjastoja, joista OpenCV on yksi esimerkki. Python-tulkki asennetaan sivustolta www.python.org

Visual Studio Code on Microsoftin ilmainen kehitysympäristö. Se on tavallaan kevytversio järeämmästä Visual Studiosta. Python-koodia (kuten mitä tahansa muutakin tekstimuotoista koodia) voi kirjoittaa toki vaikka tavallisella tekstieditorilla, mutta kehitysympäristö tuo mukanaan mukavia työkaluja kuten automaattisen täydentämisen, virheenetsinnän ja debuggerin. Visual Studio Code asennetaan osoitteesta code.visualstudio.com

Pip on Pythonin paketinhallintatyökalu. Minkä tahansa Pythonin pakettitietokannassa olevan kirjaston voi asentaa koneelleen pilvestä komennolla `pip install <paketin_nimi>`. OpenCV asennetaan komennolla `pip install opencv-python`. Kun OpenCV:n asentaa, mukana tulee myös OpenCV:n esivaatimus, numeerisen laskennan kirjasto Numpy.

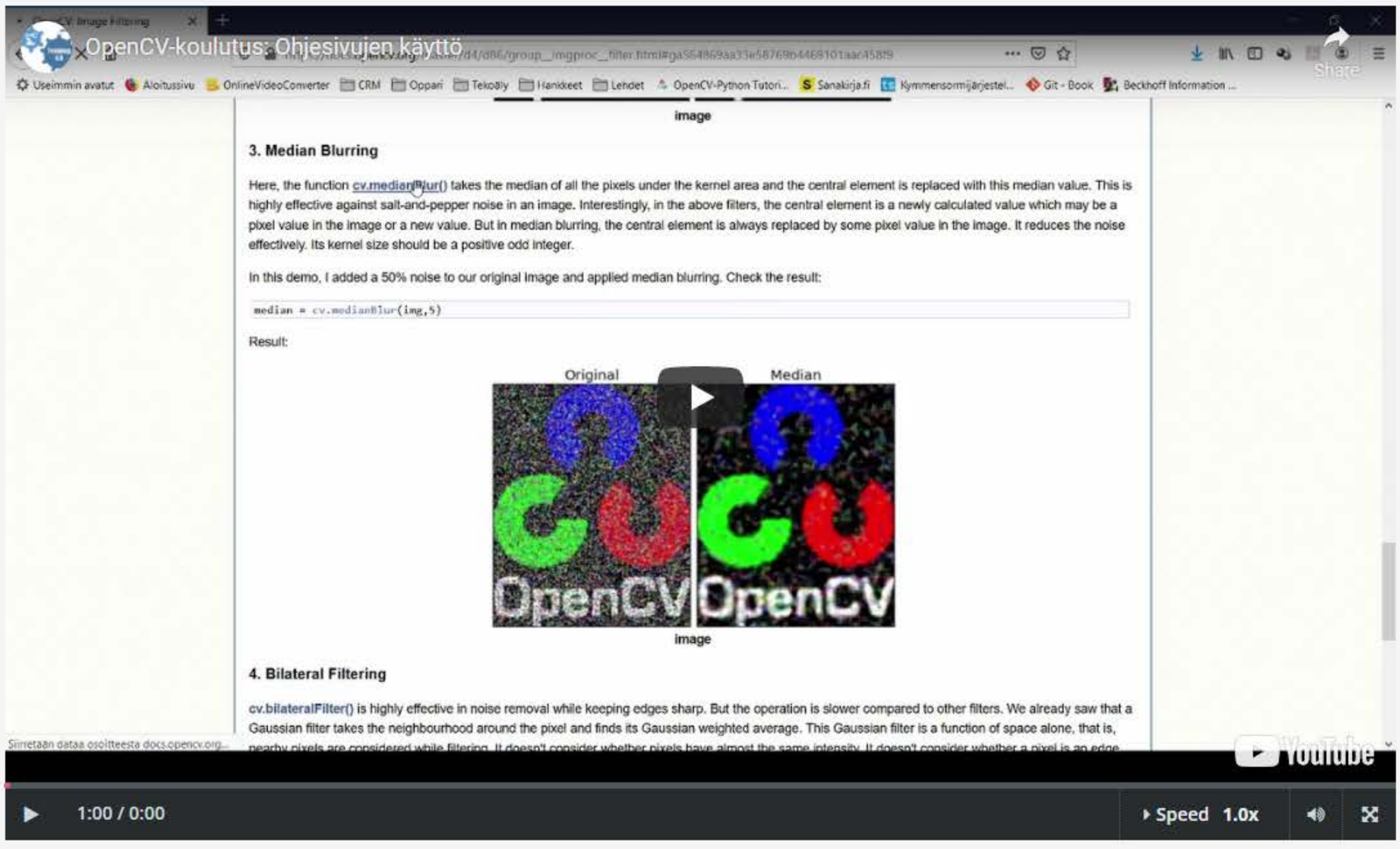
OpenCV:n dokumentaatiosivut sisältävät hyödyllisiä tutoriaaleja ja totta kai kaikkien OpenCV:n komentojen ohjesivut. Sivuston käyttö esitellään seuraavassa videossa. Ongelmatilanteisiin törmätessä tai ylipäätään apua etsiessä vanha kunnon Googlen hakukone on myös toimiva työkalu. Laajan käyttäjäyhteistön ansiosta keskustelupalstoilla kuten stackoverflow, on paljon OpenCV:hen liittyviä kysymyksiä ja vastauksia, ja yleensä haut vievät niihin.

STAFF DEBUG INFO

Dokumentaatiovideo

Bookmark this page

Dokumentaatiosivujen käyttö

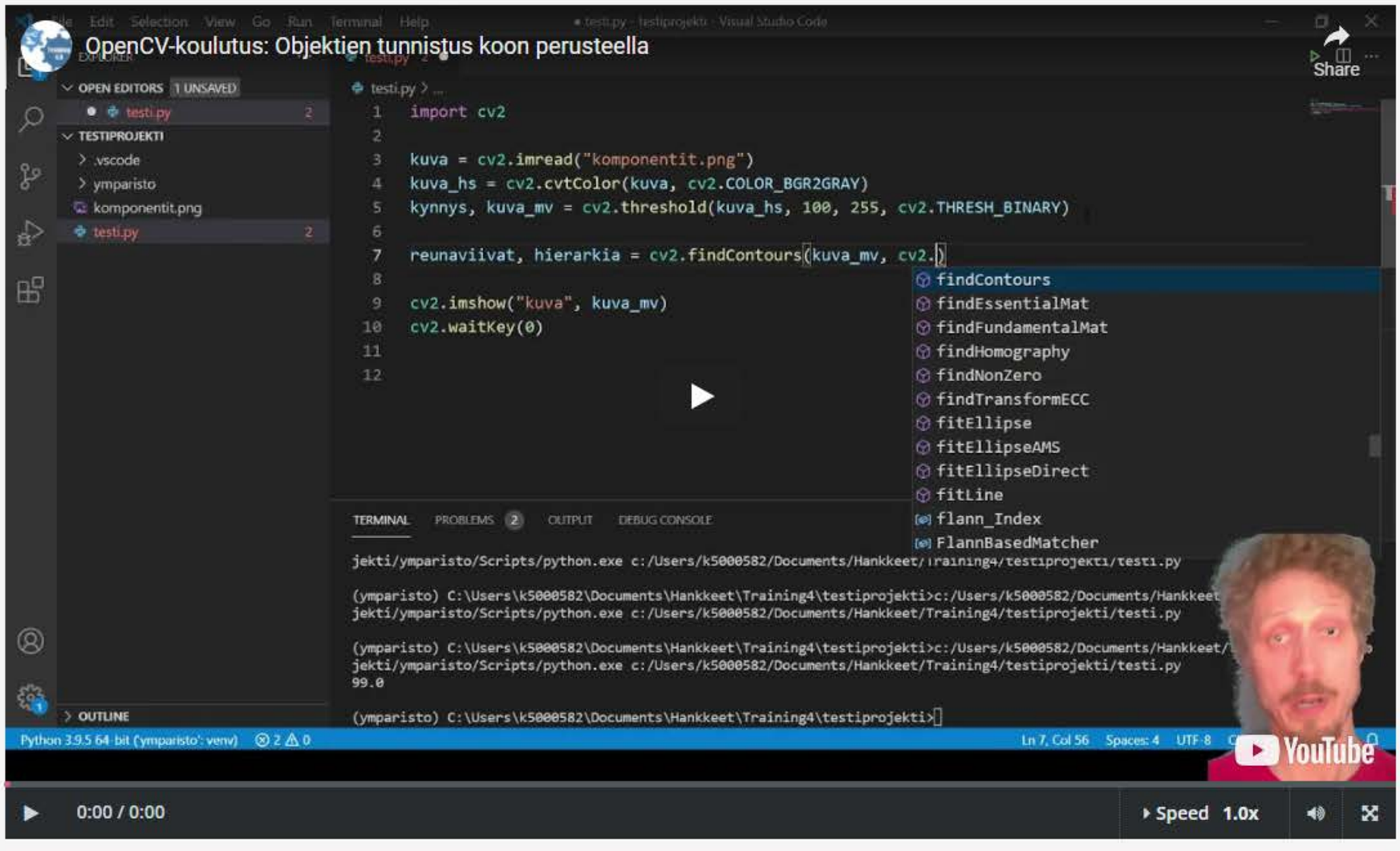


STAFF DEBUG INFO

Video: objektien tunnistus koon perusteella

Bookmark this page

Objektien tunnistaminen koon perusteella



STAFF DEBUG INFO

Videoesimerkin koodi ja lisätietoa

Bookmark this page

Videolla käsitelty koodi on alla muutamin kommentein ja lisäyksin. Python on tarkka tabuloinneista, joten tarkista, että ne menevät oikein, kun kopioit koodin. Kokeile ajaa koodia kotikoneellasi. Seuraavassa osiossa on kysymyksiä koodiin liittyen. Kopioi siis koodi Visual Studio Coden puolelle, ja testaile sitä eri muutoksilla.

Tarvittava kuva on täällä.

```
import cv2

# Luetaan kuva kovalevyltä. Jos kuva tai kuvakansio on samassa
# hakemistossa kuin kooditiedosto, riittää suhteellinen polku.
# Nyt imread lukee komponentit.png-nimisen kuvan samasta hakemistosta
# kuin missä kooditiedosto on. Jos kuvan lukeminen epäonnistuu
# (polku on virheellinen), kuvan arvoksi tulee None
kuva = cv2.imread("komponentit.png")
if kuva is None:
    print("Kuvan lukeminen epäonnistui. Tarkista kuvapolku")
else:
    # Muutetaan harmaasävykuvaksi ja edelleen mustavalkokuvaksi.
    kuva_hs = cv2.cvtColor(kuva, cv2.COLOR_BGR2GRAY)
    kynnys, kuva_mv = cv2.threshold(kuva_hs, 100, 255, cv2.THRESH_BINARY)

    # Etsitään reunaviivat
    reunaviivat, hierarkia = cv2.findContours(kuva_mv, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Tuloksena saatu muuttuja 'reunaviivat' on lista reunaviivojen x-y-
    # koordinaateista. Siinä on siis yhtä monta x-y-pistetaulukkoa kuin
    # reunaviivoja.
    # Käydään reunaviivat yksitellen läpi for-silmukassa. Lasketaan
    # jokaisen niistä pinta-ala. Jos pinta-ala ylittää 1000 neliöpikseliä,
    # piirretään reunaviiva alkuperäiseen värikuvaan punaisella.
    for reunaviiva in reunaviivat:
        if cv2.contourArea(reunaviiva) < 1000:
            cv2.drawContours(kuva, [reunaviiva], -1, (0, 0, 255), 5)

    # Näytetään lopputulos ja odotetaan käyttäjältä näppäimen painallusta.
    cv2.imshow("kuva", kuva)
    cv2.waitKey(0)
```

OpenCV:n dokumentaatiosivulla on hyvät opastukset kynnystyksestä ja ääriviivoista. Ne kannattaa lukea ajatuksella läpi.

STAFF DEBUG INFO

Harjoitus

Bookmark this page

Kysymys 1: laskutoimitukset

1 point possible (ungraded)

Kuvan lukemisen eli `imread`-rivin jälkeen kokeile laskutoimituksia kuvalla:

```
kuva_lis = cv2.add(kuva, 50)

kuva_vah = cv2.subtract(kuva, 50)
```

Mikä seuraavista on totta?

- kuva_lis on kirkkaampi kuin alkuperäinen kuva, kuva_vah on tummempi kuin alkuperäinen kuva
- kuva_lis on tummempi kuin alkuperäinen kuva, kuva_vah on kirkkaampi kuin alkuperäinen kuva
- kuva_lis on 50 riviä korkeampi kuin alkuperäinen kuva, kuva_vah on 50 riviä matalampi kuin alkuperäinen kuva
- kuva_lis on 50 saraketta leveämpi kuin alkuperäinen kuva, kuva_vah on 50 saraketta leveämpi kuin alkuperäinen kuva

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: kynnysarvot

1 point possible (ungraded)

Kokeile eri kynnysarvoja `threshold`-funktiolle ja näytä mustavalkoinen kuva kuva_mv komennolla `imshow`. Millä kynnysarvolla harmaat riviliittimet häviävät mustavalkokuvasta miltei kokonaan, mutta pienemmät muovikomponentit edelleen näkyvät?

- 210
- 220
- 240
- 255

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: lajittelu piirin avulla

1 point possible (ungraded)

Kokeile lajitella komponentit pinta-alan sijaan piirin avulla. Funktio on `arcLength`. Katso mitä parametreja se vaatii. Millä piirille asetetulla ehdolla ainoastaan riviliitinten reunaviiva näkyy korostettuna, kun kynnysarvo on 100?

- > 200
- > 500
- > 800
- > 1000

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: rajaavat suorakulmiot

1 point possible (ungraded)

Tutustu rajaavien suorakulmioiden laskemiseen ja piirtoon täällä, kohta 7. Laske riviliittimille pienimmän pinta-alan rajaavat suorakulmiot (rotated rectangle) ja piirrä ne kuvaan. Suorakulmioden keskipisteet (kun kynnystyksessä käytetään kynnystä 100) ovat yhteen desimaaliin pyöristettynä

Huom! Jotta voit käyttää linkin esimerkissä käytettyä komentoa int0, sinun pitää ottaa käyttöön myös numpy-kirjasto komennolla `import numpy as np` tiedoston alussa

- (352,4, 482,0), (636,5, 416,5) ja (256,3, 248,4)
- (126,4, 154,9), (125,4, 154,5) ja (154,8, 124,4)
- (282,3, 643,1), (152,1, 128,4) ja (712,4, 365,9)
- (443,1, 300,0), (664,4, 305,7) ja (482,5, 206,2)

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: muotojen vertailu toisiinsa

1 point possible (ungraded)

Tutustu funktion `matchShapes` toimintaan täällä.

Käytä oheista kuvatiedostoa kuvana, johon vertailu tehdään. Muuta esimerkkikoodin toimintaa niin, että punaiset ääriviivat piirretään vain kyseistä muoviosaa vastaaville osille. Mitä lukua pienempi matchShapesin tuloksen pitää olla, että tämä onnistuu??

- 0.001
- 0.01
- 0.1
- 1

Show answer

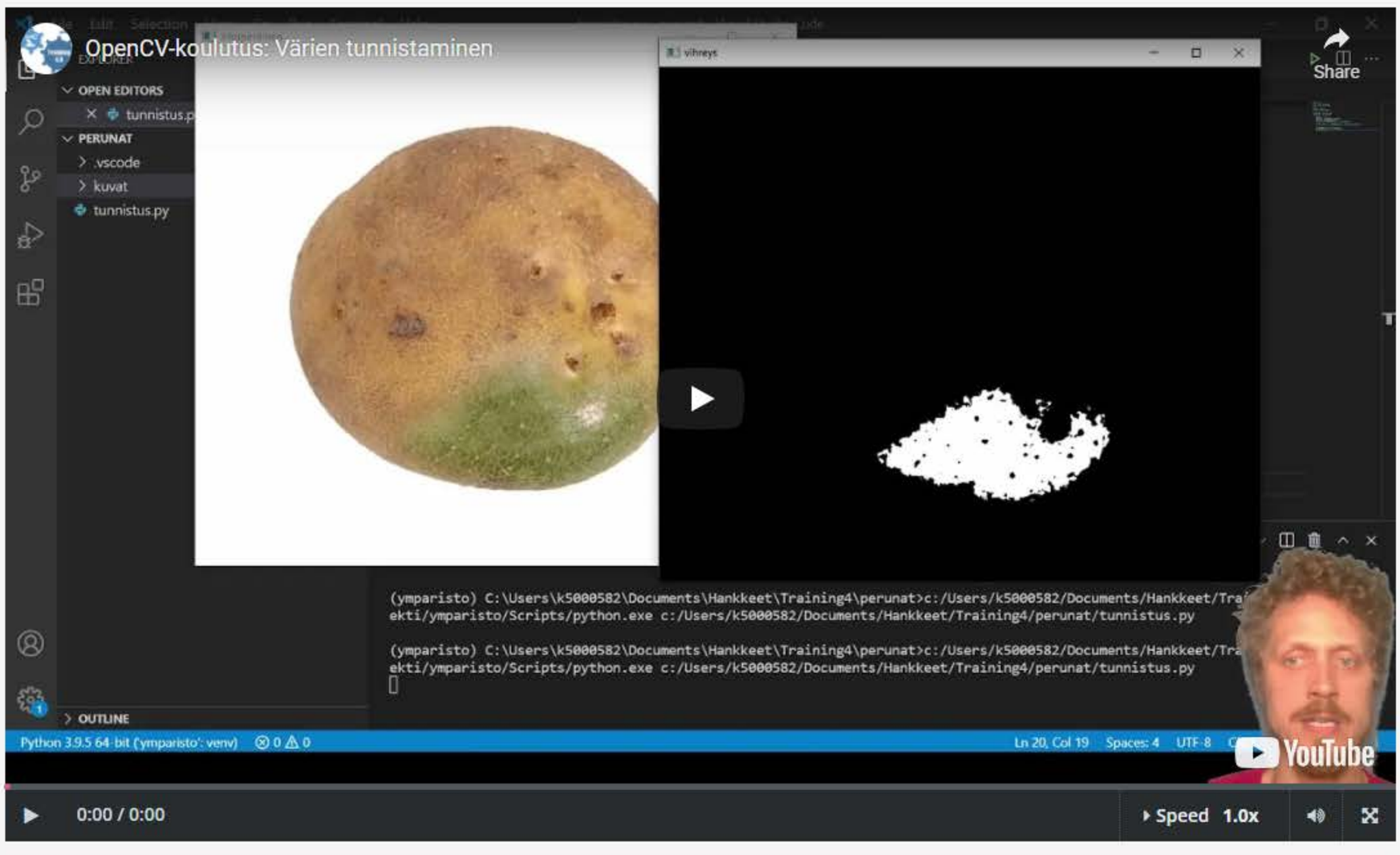
Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Video: värikuvat ja värien tunnistaminen

Bookmark this page

Värien tunnistaminen



STAFF DEBUG INFO

Videoesimerkin koodi ja lisätietoa

Bookmark this page

Videolla kirjoitettu ja läpi käyty koodi on alla muutamin kommentein varustettuna. Kopioi se taas omalle koneellesi ja koita saada toimimaan. Kuvasetti on saatavilla täällä. Osion harjoituksissa koodiin taas tehdään muutoksia ja kokeiluita.

Väreistä ja väriavaruuden muuttamisesta on OpenCV:n dokumentaatiosivuilla opastus. Se kannattaa taas lukea. Opastuksen esimerkissä käytetään koneen webbikameraa sinisen objektin seuraamiseen. Vastaava esimerkki on tämän kurssin osiossa *kuvan lukeminen kameralta*.

```
import cv2
from os import listdir
from os.path import join

# Kuvakansio ja sen tiedostolistaus
kansio = "kuvat"
lista = listdir(kansio)

# Vihreän värin ylä- ja alarajojen HSV-arvot
vihrea_yla = (35, 255, 255)
vihrea_ala = (25, 50, 50)

# Käsitellään tiedostolistassa olevia tiedostonimiä
# yksi kerrallaan, muodostetaan niistä yhdessä kansion
# polun kanssa kokonainen kuvapolku ja avataan näin kuvia
# yksi kerrallaan. Tehdään kuville HSV-muunnos.
for tiedosto in lista:
    kuvapolku = join(kansio, tiedosto)
    kuva = cv2.imread(kuvapolku)
    kuva_hsv = cv2.cvtColor(kuva, cv2.COLOR_BGR2HSV)

    # Tuotetaan kuvasta mustavalkokuva, jossa valkoisella näkyy ainoastaan
    # alue, jossa on vihreää. Etsitään näiden alueiden reunaviivat ja piirretään
    # ne punaisella.
    vihreat_alueet = cv2.inRange(kuva_hsv, vihrea_ala, vihrea_yla)
    reunaviivat, _ = cv2.findContours(vihreat_alueet, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cv2.drawContours(kuva, reunaviivat, -1, (0, 0, 255), 3)

    cv2.imshow("alkuperainen", kuva)
    cv2.waitKey(0)
```

Harjoitus

Bookmark this page

Kysymys 1: reunaviivojen karsiminen

1 point possible (ungraded)

You can use this template as a guide to the simple editor markdown and OLX markup to use for numerical input problems. Edit this component to replace this template with your own assessment.

Add the question text, or prompt, here. This text is required.
You can add an optional tip or note related to the prompt like this.

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: valkoinen väri

1 point possible (ungraded)

Jos haluat perunakuvista kaiken muun, paitsi valkoisen taustan (ja jotain varjoja), mille HSV-kuvan kanavalle sinun täytyy asettaa rajat (muut voivat olla minimistä maksimiin)?

- H:lle
- S:lle
- V:lle

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: valkoinen väri

1 point possible (ungraded)

Mikä alaraja tälle kanavalle täytyy silloin asettaa?

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: valkoinen väri

1 point possible (ungraded)

Entä yläraja?

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: ruskea väri

1 point possible (ungraded)

Pida S:n ja V:n rajat samoina kuin osion esimerkissä (50 - 255). Millä H:n välillä löydät parhaiten perunoiden ruskeat alueen?

- 0 - 5
- 5 - 10
- 10 - 15
- 15 - 20

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 6: HSV:n tulkinta BGR:nä

1 point possible (ungraded)

Jos näytät `imshow`-komennolla kuvan `kuva_hsv`, värit menevät sekaisin, sillä OpenCV odottaa BGR-kuvaa ja tulkitsee näin HSV-kuvan kanavat BGR-kuvan värikanavina. Minkä värinen (suurimmaksi osaksi) on nyt perunakuvien tausta näytöllä näkyvissä kuvissa?

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Monimutkaisemmat kuvaoperaatiot: johdanto

Bookmark this page

Kuvan kynnystäminen mustavalkoiseksi joko harmaasävyn tai värin perusteella ja näin saatujen yhtenäisten valkoisten osioiden ominaisuuksien laskenta ja vertailu reunaviivoja (contours) ja niille rakennettuja funktioita hyödyntämällä on hyvin tehokas ja käytetty strategia moniin kohteiden geometriaan perustuviin konenäköongelmiin. Lisäksi voidaan tarvita esikäsittelynä (ennen kynnystystä) jonkinlaista suodatusta kuten pehmennystä, joka poistaa kohinaa ja häivyttää yksityiskohtia, tai reunanetsintää, joka korostaa kuvassa näkyviä reunoja, ja jälkikäsittelynä (kynnystyksen jälkeen) erilaisia morfologisia operaatioita, jotka siistivät saatua mustavalkoista kuvaa. Hyvän valaistuksen käyttö vähentää esikäsittelyn ja jälkikäsittelyn tarvetta.

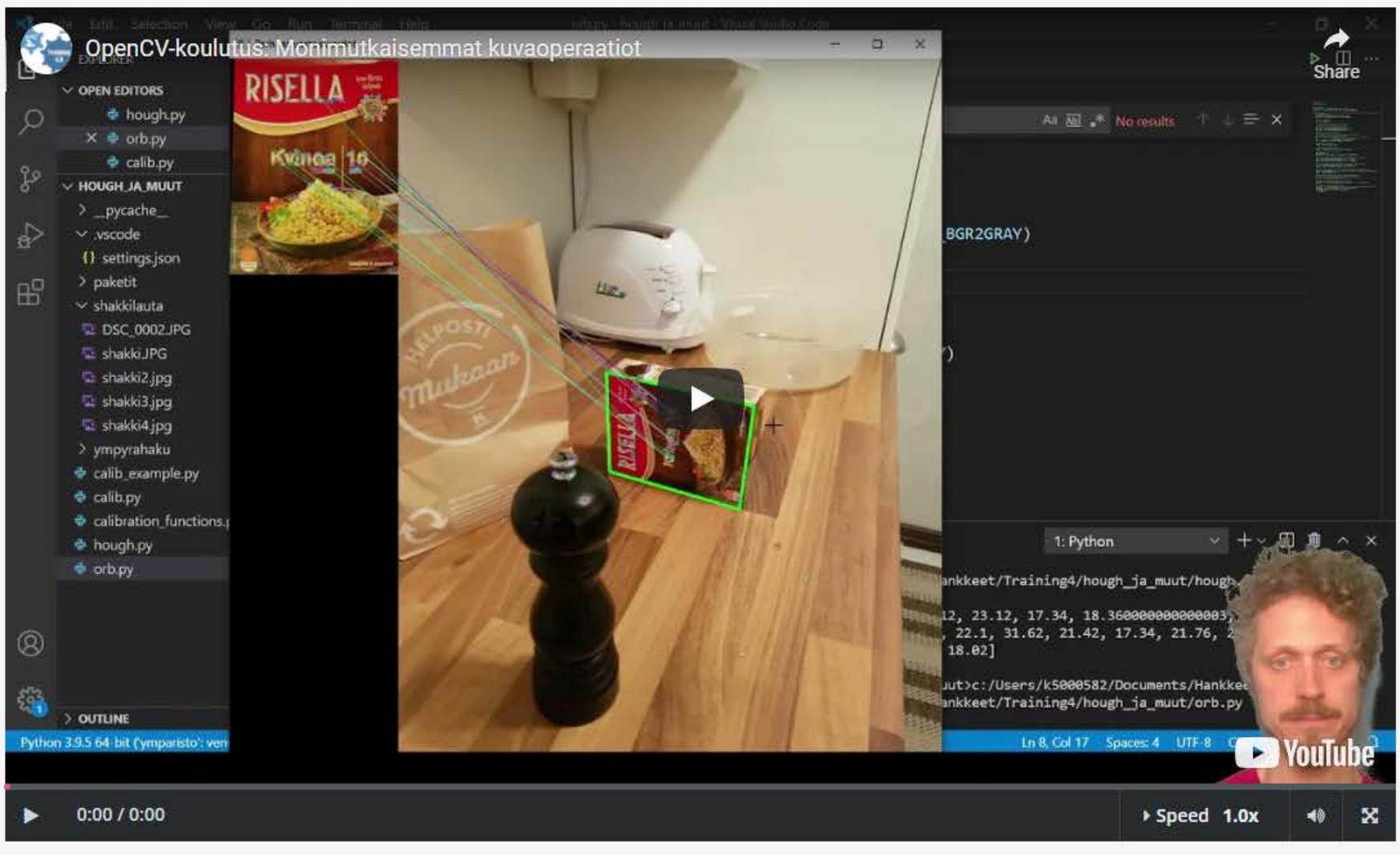
Kynnystämiseen ja alueiden etsintään perustuvien tarkastusrutiinien lisäksi OpenCV:llä pystyy tekemään paljon muutakin. Tämän osion videossa esitellään kolme esimerkkiä: ympyrämäisten muotojen hakeminen, kuvan piirteiden laskeminen ja niihin pohjautuva vastaavuuksien etsiminen kahdesta kuvasta sekä linssivääristymän poisto shakkimaista ruudukkoa käyttämällä. Koska nämä vaativat jo huomattavasti enemmän koodirivejä, ohjelmakoodeja ei käydä videolla juurikaan läpi, vaan videolla keskitytään näyttämään sovellusesimerkkejä. Ympyrähaun ja piirrehaun koodit kuitenkin esitellään tekstiosuuksissa ja niistä molemmista on harjoitukset. Linssivirheen poistosta ja kameran kalibroinnista kannattaa lukea niitä käsittelevä osuus OpenCV:n dokumentaatiosivuilta.

STAFF DEBUG INFO

Video: monimutkaisemmat kuvaoperaatiot

Bookmark this page

Monimutkaisemmat kuvaoperaatiot



STAFF DEBUG INFO

Ympyrähaun koodi ja lisätietoa

Bookmark this page

Alla koodi, jolla ruuvin kannat etsitään piirilevystä. Molemmat ympyrähaun kuvat ovat ladattavissa täältä. Ympyrähaussa käytettävästä Houghin ympyrämuunnoksesta on lyhyt esittely OpenCV:n dokumentaatiosivuilla. Tutustu myös itse funktion `HoughCircles` dokumentaatioon.

Funktiolle `HoughCircles` kerrotaan, miten vahvoja ympyrämäisiä muotoja halutaan etsiä, kuinka monta pistettä niiden kehälle tulisi löytyä (ajattele jonkinlaisena yhdistä pisteet -tehtävänä) ja millä välillä ympyröiden halkaisijan tulisi olla. Houghin ympyrähaku käyttää gradienttimenetelmää etsimään kuvasta reunoja ja sitten yrittää tavallaan piirtää ympyröitä kuvassa oleviin edellä annetut kriteerit täyttäviin reunoihin. Reunojen etsinnästä on myös esittely OpenCV:n dokumentaatiosivuilla, kuten tämän osion johdannossa mainittiin. Ennen reunojen etsintää kuvaa pehmennetään, jolloin vähennetään sen yksityiskohtien määrää (yhdistä pisteet -tehtävien pisteiden määrää). Pehmennyksen tarkoitus tässä on siis häivyttää muotoja, jolloin Hough-muunnokselle jää vähemmän potentiaalisia piirteitä, johon se yrittää ympyrää piirtää. Tässä oletetaan, että ympyrämäiset piirteet ovat kuvassa sen verran selkeitä, että ne eivät ole ensimmäisiä, jotka pehmennys häivyttää.

Tukkien halkaisijan laskemisessa käytetään suurin piirtein samaa koodinpätkää, mutta sen parametreja täytyy muuttaa. Tätä katsotaan osion harjoituksissa.

Hieman toisenlaista Hough-muunnosta käytetään suorien viivojen hakuun. Myös sen esittely kannattaa lukea.

```
import cv2
import numpy as np

# Luetaan kuva värillisenä tuloskuvan piirtämistä varten.
# Muutetaan harmaasävykuvaksi Hough-muunnosta varten.
kuva = cv2.imread("ympyrahaku/piirilevy.jpg")
kuva_hs = cv2.cvtColor(kuva, cv2.COLOR_BGR2GRAY)

# Pehmennetään, jotta vähennetään yksityiskohtia.
kuva_hs = cv2.medianBlur(kuva_hs, 5)

# Etsitään ympyrät
ympyrat = cv2.HoughCircles(kuva_hs, cv2.HOUGH_GRADIENT, 1, 50, param1=50, param2=30, minRadius=15, maxRadius=20)

# Tulostetaan määrä
print(f"Ruuveja on yhteensä {len(ympyrat[0])}")

# Pyöristetään ympyröiden keskipisteiden koordinaatit ja säteet
# kokonaisluvuiksi ja piirretään
ympyrat = np.uint16(np.around(ympyrat))

for y in ympyrat[0]:
    cv2.circle(kuva, (y[0], y[1]), y[2], (255, 0, 0), 2)
    cv2.circle(kuva, (y[0], y[1]), 2, (0, 0, 255), 3)

cv2.imshow('detected ympyrat', kuva)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Harjoitus: ympyrähaku

Bookmark this page

Kysymys 1: Houghin ympyrämuunnoksen palatusarvot

1 point possible (ungraded)

Funktio `HougCircles` palauttaa jokaista ympyrää kohden kolme lukua. Mitä nämä ovat? Erittele vastauksesi pilkuilla tyyliin *pinta-ala, ympärysmitta, väri* .

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: Houghin ympyrämuunnoksen parametrit

1 point possible (ungraded)

Mitä ovat funktion `HoughCircles` parameterit `param1` ja `param2`, kun hakutapa on `HOUGH_GRADIENT`?

- `param1` on ympyrän minimisäde ja `param2` maksimisäde
- `param1` liittyy reunantunnistuksen herkkyyteen ja `param2` liittyy ympyrähaun herkkyyteen
- `param1` on kynnystysfunktion kynnys ja `param2` ympyrän kehälle osuvien pisteiden minimimäärä
- `param1` liittyy pehmennyksen vahvuuteen ja `param2` keskipisteiden etäisyyteen toisistaan

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: Houghin ympyrämuunnos ja uusi kuva

1 point possible (ungraded)

Jos kokeilet osiossa annettua ympyrähaun esimerkkikoodia suoraan kuvaan tukit.jpg, montako ympyrää löytyy?

- 0
- 8
- 26
- 127

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: Houghin ympyrämuunnoksen parametrien säätäminen

2 points possible (ungraded)

Käytä kuvaa tukit.jpg osion ympyrämuunnoksen esimerkkikoodille. Muuta pehmennysfunktion `medianBlur` toinen parametri 5:stä 11:een. Pidä funktion `HoughCircles` parametrit muuten samoina, mutta muuta parametreja `minRadius` ja `maxRadius`. Millä arvoilla tukit löytyvät hyvin (aivan kaikki eivät löydy, sillä osa on aika soikeita) eikä saman tukin ympärille piirry kahta ympyrää? Anna vastauksesi viiden tarkkuudella (esim. 15 tai 20 tai 25...). Ensimmäiseen kenttään `minRadius` ja toiseen `maxRadius`

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: Tukkien halkaisijoiden laskeminen sentteinä

1 point possible (ungraded)

Halkaisijan laskemisen jälkeen sen saa kirjoitettua kuvaan käyttämällä komentoa `putText`. Kuvassa oletetaan yhden pikselin vastaava 1,7 millimetriä. Kuinka tukin halkaisija senttimetreinä lasketaan ympyrämuunnoksen tietojen avulla esimerkkikoodin `for`-silmukassa?

- y[1] * 2 * 1.7
- y[2] * 2 * 1.7
- y[1] * 2 * 0.17
- y[2] * 2 * 0.17
- y[1] * 2 / 0.17
- y[2] * 2 / 0.17
- y[1] * 2 * pi / 1.7
- y[2] * 2 * pi / 1.7

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Piirrehaun koodi ja lisätietoa

Bookmark this page

Yleinen konenäköongelma on etsiä kuvasta jokin tietty esine tai asia. Yksinkertaisin tapa tietyn asian löytämiseen kuvasta on niin sanottu hahmon vertailu (template matching). Hahmon vertailussa etsitään isommasta kuvasta pienemmän kuvan esittämää aluetta. Alueen tosin pitää olla samankokoinen ja samassa orientaatiossa kuin pienemmässä kuvassa. Hahmon vertailusta on esittely OpenCV:n dokumentaatiosivuilla.

Hyvin usein vähintään kohteen orientaatiota ei voida tietää etukäteen. Esimerkiksi etsittäessä alustalta tiettyä osaa, osa voi olla satunnaisessa asennossa. Lisäksi osa ei välttämättä näy kokonaan, vaan se on osittain peitossa. Joissain tapauksissa osaa myös katsotaan eri kuvakulmasta kuin mistä mallikuva on kuvattu. Tällaisissa tapauksissa hahmon vertailu toimii hyvin huonosti. Osa voidaan kuitenkin löytää piirteitä hyödyntämällä.

Piirteet ovat laskennallisia "helposti löydettäviä alueita" kuvassa. Ne voivat siis kuvata esimerkiksi kulmaa, reunaa tai äkillistä vaihdosta kirkkaudessa. Piirteitä etsimään on kehitetty erityisiä piirrekuvaajia (feature descriptor), jotka huolehtivat siitä, että haku ei ole herkkä mittakaavan muutoksille. Tunnetuimpia piirrekuvaajia ovat patentoidut SIFT ja SURF, mutta videon esimerkissä käytetään ORBia, joka on avoimen lähdekoodin piirrekuvaaja. Piirteistä on pidempi osio OpenCV:n dokumentaatiosivuilla.

Alla on videolla nähdyn esimerkin ohjelmakoodi kommentoituna. Esimerkissä tarvittavat kuvat löydät täältä. Alussa luetaan kuva etsittävästä kohteesta ja alueesta, josta kohde pitäisi löytää (etsintäkuva). Sitten luodaan ORB-etsin, jonka avulla etsitään kummastakin kuvasta hyvät laskennalliset piirteet. Funktio `detectAndCompute` palauttaa avainpisteet, joissa on tieto piirteiden sijainnista kuvassa sekä niiden kiertymästä, ja piirteen kuvaajan, joka ORB:n tapauksessa käsittää 32 8-bittistä lukua per piirre. Tätä voi ajatella jonkinlaisena piirteen sormenjälkenä - mitä samankaltaisempi piirre, sitä samankaltaisemmat nämä luvut ovat. Seuraavaksi koodiesimerkissä luodaan brute force -vertailin, jonka avulla eniten toisiaan vastaavat piirteet kuvien välillä paritetaan keskenään ja aikaansaadut piirreparit järjestetään paremmuusjärjestykseen sillä perusteella, kuinka hyvin ne vastasivat toisiaan. Nyt suurin osa listan alkupään pistepareista on luultavasti oikeita vastinpisteitä - kuten videolla näkyy, mukaan jää myös virheellisiä pareja (näet tämän itsekin, kun ajat koodin: tuloskuvaan piirtyy vastaavuutta kuvaavia viivoja virheellisten pisteparien välille). Parhaiten toisiaan vastaavien piirteiden x-y-koordinaattien perusteella tehdään perspektiivimuunnos - halutaan siis tietää, missä kuvakulmassa etsittävä kohde on etsintäkuvassa. Kun perspektiivimuunnos on laskettu, etsittävän kohteen kokoinen suorakulmio muunnetaan löytynyttä kohdetta vastaavaan perspektiiviin etsintäkuvassa ja piirretään etsintäkuvaan. Jos kaikki on onnistunut, nelikulmion tulisi nyt rajata etsitty kohde.

```python
import cv2
import numpy as np

# Mitä etsitään
kuva_malli_bgr = cv2.imread("paketit/paketti.jpg")
kuva_malli = cv2.cvtColor(kuva_malli_bgr, cv2.COLOR_BGR2GRAY)

# Mistä etsitään (for-silmukalla käydään läpi kuvat poyta1.jpg...poyta4.jpg)
for i in range(1, 4):
    kuva_bgr = cv2.imread(f"paketit/poyta{i}.jpg")
    kuva = cv2.cvtColor(kuva_bgr, cv2.COLOR_BGR2GRAY)

    # Luodaan ORB-etsin
    orb = cv2.ORB_create()

    # Etsitään piirteet (toinen parametri maskille)
    # kp: keypoint: paikka, kiertymä, skaalus yms. tieto
    # des: descriptor: kuvailin - visuaalinen "sormenjälki"
    kp1, des1 = orb.detectAndCompute(kuva_malli, None)
    kp2, des2 = orb.detectAndCompute(kuva, None)

    # Tehdään brute force -vertailin
    ## ensimmäinen parametri on käytettävä normalisointi-ikkuna
    ##     Hamming on vähän Gaussihtava käppyrä
    ## Toinen parametri: ristitsekkaus, jos totta, otetaan vain lailliset parit
    ##                   eli tasan kahden pisteen muodostamat
    bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)

    # Käytetään vertailinta
    osumat = bf.match(des1, des2)

    # Laitetaan suuruusjärjestykseen
    osumat = sorted(osumat, key=lambda x: x.distance)

    # Otetaan vain tietyt etäisyydet
    MAX_ETAISYYS = 60
    hyvat = [o for o in osumat if o.distance <= MAX_ETAISYYS]

    # Etsitään hyviä piirteitä vastaavien avainpisteiden x-y-koordinaatit
    src_p = np.float32([kp1[h.queryIdx].pt for h in hyvat])
    dst_p = np.float32([kp2[h.trainIdx].pt for h in hyvat])

    # Perspektiivimuunnos mallipisteistä (pieni kuva) ison kuvan pisteisiin
    M, _ = cv2.findHomography(src_p, dst_p, cv2.RANSAC, 5.0)
    korkeus, leveys = kuva_malli.shape # Mallikuvan mitat

    # Mallikuvan ääripisteet (reshapella lisätään OpenCV:n vaatima ylimääräinen hakasulku 2D-koordinaatteihin)
    kulmat = np.float32([ [0, 0], [0, korkeus-1], [leveys-1, korkeus-1], [leveys-1, 0]])
    kulmat = kulmat.reshape(-1, 1, 2)

    # Muunnetaan ääripisteet saadulla perspektiivimuunnoksella:
    # Tässä siis mallikuva on isommassa kuvassa ja tässä asennossa.
    dst = cv2.perspectiveTransform(kulmat, M)

    # Rajataan viivoilla pikkukuvan sijainti isossa
    kuva_bgr = cv2.polylines(kuva_bgr, [np.int32(dst)], True, (0, 255, 0), 5)

    # Piirretään lisäksi 10 parasta osumaa
    tuloskuva = cv2.drawMatches(kuva_malli_bgr, kp1, kuva_bgr, kp2, osumat[:20], None)
    tuloskuva = cv2.resize(tuloskuva, None, fx=0.5, fy=0.5)

    cv2.imshow("Parhaat vastaavuudet", tuloskuva)
    cv2.waitKey(0)
```

Harjoitus: piirrehaku

Bookmark this page

Kysymys 1: etsittävän kohteen reunojen näyttäminen tuloskuvassa

1 point possible (ungraded)

Miten on mahdollista, että paketin reunat piirtyvät oikein kuvassa poyta2.jpg, vaikka paketti on osin peitetty?

- paperipussi on osin läpinäkyvää materiaalia
- piirteet hyödyntävät koneoppimista, ja tarpeeksi monta pakettimaista kuvaa nähtyään piirre-etsin pystyy päättelemään kohteen muodon
- näkyvästä osuudesta löytyy tarpeeksi monta piirrettä siihen, että paketin muoto kuvassa pystytään päättelemään
- piirre-etsin hyödyntää tässä kuvia poyta1.jpg ja poyta3.jpg

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: RANSAC-menetelmä

1 point possible (ungraded)

Esimerkkikoodin rivi

```
M, _ = cv2.findHomography(src_p, dst_p, cv2.RANSAC, 5.0)
```

etsii perspektiivimuunnoksen kuvista löytyneiden piirteiden koordinaattien vastinpisteiden välille. Eli pikkukuvan piirteiden koordinaattien ja ison kuvan piirteiden koordinaattien välille. Idea on siis etsiä, missä kuvakulmassa etsittävä "pikkukuva" on isossa kuvassa. Yleensä perspektiivimuunnos etsitään yksinkertaisesti komennolla

```
M, _ = cv2.findHomography(src_p, dst_p)
```

Voit kokeilla muuttaa koodin näin. Huomaat, että enää tuloskuvissa ei enää näy etsittävän kohteen rajausta isossa kuvassa. Kaksi ylimääräistä parametria siis tarvitaan. Hyvä esitelmä RANSAC-menetelmästä on täällä. Plärää kalvot läpi ja vastaa sen perusteella, miksi RANSAC-menetelmää tarvitaan perspektiivimuunnoksen etsimisessä tässä esimerkissä.

- Se nopeuttaa huomattavasti muunnoksen laskemista. Ilman RANSAC-menetelmää muunnos on suuren pisteparimäärän takia niin hidas laskea, että lasku lopetetaan ennenaikaisesti.
- Koordinaattien joukossa on niin monta pistettä, jotka eivät oikeasti vastaa toisiaan. Ilman RANSAC-menetelmän käyttöä perspektiivimuunnoksen lasku menee näiden takia väärin.
- Ilman RANSAC-menetelmää perspektiivimuunnoksessa otetaan huomioon vain kiertymä, siirtymä ja skaala, muttei muita vääristymiä.

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: toisen kuvapakan kokeileminen

1 point possible (ungraded)

You can use this template as a guide to the simple editor markdown and OLX markup to use for multiple choice problems. Edit this component to replace this template with your own assessment.

Kokeile esimerkin koodia oheisen paketin kuville. Jotta kaikki kaappi-kuvat käsitellään, muuta `for`-silmukka rullaamaan välin `(1, 5)`. Mistä kuvista cd löytyy?

- [ ] kaappi1.jpg
- [ ] kaappi2.jpg
- [ ] kaappi3.jpg
- [ ] kaappi4.jpg

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: RANSAC-menetelmän maksimietäisyys

1 point possible (ungraded)

Käytä edelleen kuvapaketin cdt kuvia. Muuta rivin

```
M, _ = cv2.findHomography(src_p, dst_p, cv2.RANSAC, 5.0)
```

toisen parametrin lukuarvoa. Se määrää, kuinka paljon vastinpisteiden koordinaateissa saa olla virhettä että ne edelleen lasketaan vastinpisteiksi. Laske sitä yhden välein, sitten nosta sitä yhden välein. Tarkastele tuloksia. Millä arvolla cd ei enää löydy kuvasta kaappi4.jpg?

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: oma testi

1 point possible (ungraded)

You can use this template as a guide to the simple editor markdown and OLX markup to use for multiple choice problems. Edit this component to replace this template with your own assessment.

Kokeile piirrehakuesimerkkiä omalle kuvasarjallesi. Kokeile vaihdella parametria `MAX_ETAISYYS` ja RANSAC-menetelmän maksimivirhettä (viime kysymyksessä vaihdeltu parametri).

- Kokeilin
- En kokeillut

Show answer

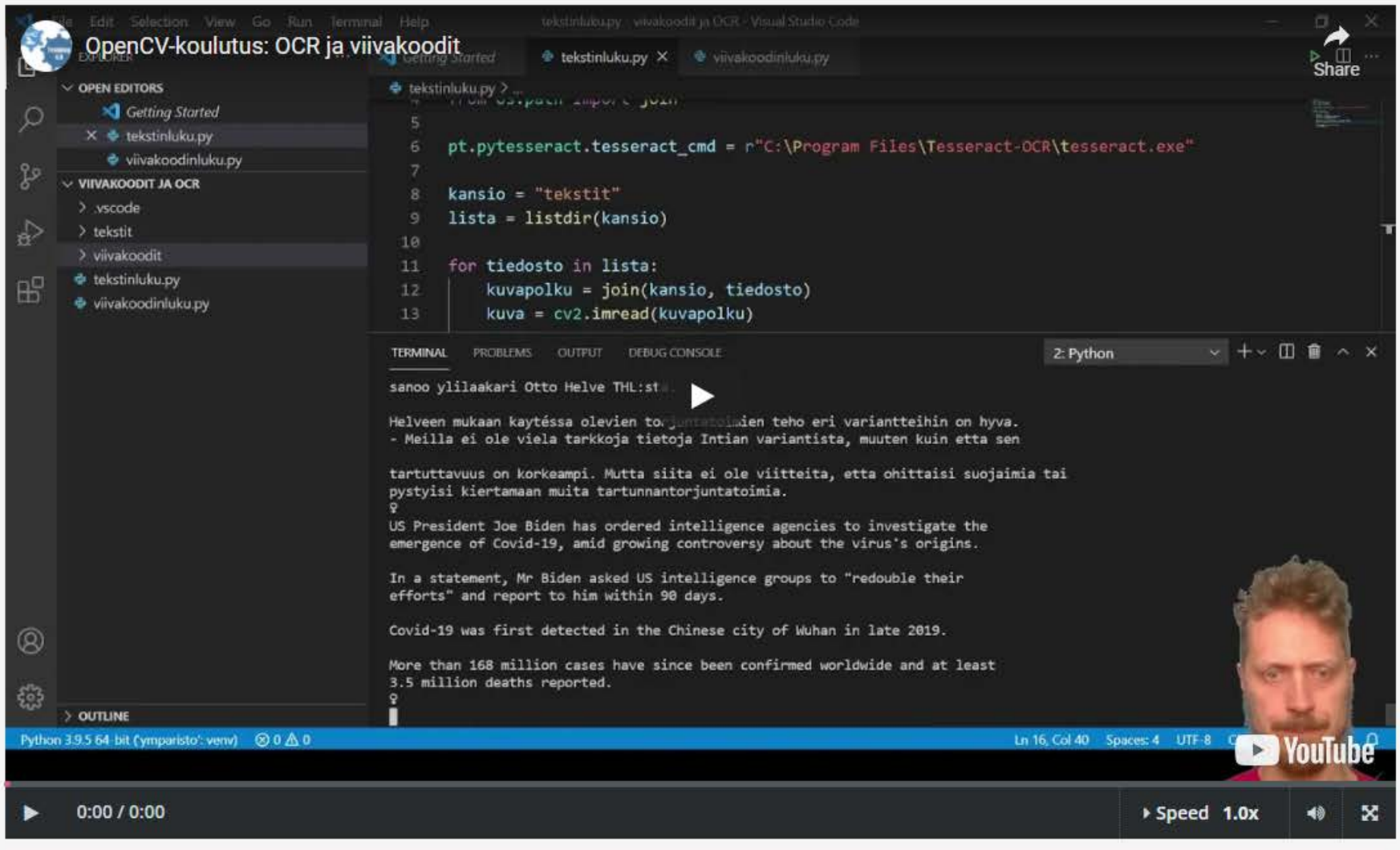
Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Koodivideo

Bookmark this page

OCR ja viivakoodit



STAFF DEBUG INFO

Videoesimerkin tekstin lukemisen koodi ja lisätietoa

Bookmark this page

Tesseract on alkujaan HP:n kehittämä ja sittemmin Googlen ylläpitämä testintunnistusohjelma (OCR), joka on kenen tahansa vapaasti asennettavissa ja käytettävissä. On epäilyjä, että Google käyttäisi sitä myös kääntäjänsä puhelinsovelluksessa, mutta se ei ole varmaa tietoa. Tesseract on itsenäinen ohjelma, mutta Python-kääreluokan avulla sitä kutsua Python-koodissa. Tesseractin hyvä puoli on se, ettei etsittävää fonttia tai sen sijaintia kuvassa tarvitse määritellä etukäteen. Toki kuvan rajaaminen usein auttaa tekstin löytymiseen, kuten video osoitti, mutta läheskään aina se ei ole tarpeen.

Tesseractin asentaminen Windowsille on helpointa osoitteesta https://github.com/UB-Mannheim/tesseract/wiki. Valitse oikea asennuspaketti 32- tai 64-bittiselle Windowsille otsikon *Tesseract installer for Windows* alta.

Jos asennat Tesseractin kaikille käyttäjille, `tesseract.exe`n polku on `C:\Program Files\Tesseract-OCR\tesseract.exe`. Jos asennat Tesseractin vain yksittäiselle käyttäjälle, polku on `C:\Users\<käyttäjätunnus>\AppData\Local\Programs\Tesseract-OCR\tesseract.exe`. Tämä polku tarvitsee kirjoittaa ohjelmakoodiin, jotta Tesseractin Python-kääreluokka pystyy ohjelmaa käyttämään. Python-kääreluokka taas asennetaan pipin kautta komennolla `pip install pytesseract`.

Alla on videolla käsitelty koodi kommentoituna ja täydennettynä niin, että se lisäksi ympäröi luetut sanat suorakulmioilla ja kirjoittaa tekstin viereen. Tarvittavat kuvat löydät täältä.

```
from pytesseract import Output
import pytesseract as pt
import cv2
from os import listdir
from os.path import join

# Tesseractin asennuspolku
pt.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
MIN_VARMUUS = 0 # Tekstiltä vaadittava minimivarmuus (0...100)

# Testikuvien kansio
kansio = "tekstit"
lista = listdir(kansio)

# Käydään kansion tiedosto tläpi yksi kerrallaan
for tiedosto in lista:
    kuvapolku = join(kansio, tiedosto)
    kuva = cv2.imread(kuvapolku)

    # Muutetaan RGB-muotoon (Tesseractin vaatimus) ja luetaan teksti
    kuva_rgb = cv2.cvtColor(kuva, cv2.COLOR_BGR2RGB)
    print(pt.image_to_string(kuva_rgb))
    tulokset = pt.image_to_data(kuva_rgb, output_type=Output.DICT)

    # Käsitellään silmukalla jokainen löydetty tekstin paikka erikseen
    for i in range(0, len(tulokset["text"])):

        # Tekstiä rajaavan suorakulmion koordinaatit
        x = tulokset["left"][i]
        y = tulokset["top"][i]
        leveys = tulokset["width"][i]
        korkeus = tulokset["height"][i]

        # Itse teksti ja varmuus sille, että kyseessä on tekstiä
        teksti = tulokset["text"][i]
        varmuus = float(tulokset["conf"][i])

        # Suodatetaan pois tulokset, joissa varmuus on alle halutun
        # (muuta siis parametria MIN_VARMUUS)
        if varmuus > MIN_VARMUUS:
            # Poistetaan teksti, joka ei ole ASCII-muotoista, jotta teksti
            # voidaan kirjoittaa OpenCV:llä kuvan päälle. Piirretään myös
            # tekstiä rajaava suorakulmio kuvaan.
            teksti = "".join([c if ord(c) < 128 else "" for c in teksti]).strip()
            cv2.rectangle(kuva, (x, y), (x + leveys, y + korkeus), (0, 255, 0), 2)
            cv2.putText(kuva, teksti, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX,
                0.5, (0, 0, 255), 2)
    cv2.imshow("Kuva ja tekstit", kuva)
    cv2.waitKey(0)
cv2.destroyAllWindows
```

Videoesimerkin viivakoodin lukemisen koodi ja lisätietoa

Bookmark this page

Viivakoodin lukemisessa tarvittava kolmannen osapuolen kirjasto `pyzbar` asennetaan pipin avulla komennolla `pip install pyzbar`. Kyseisellä projektilla on sivut PyPI-tietokannassa, ja kiinnostuneet voivat käydä niitä katsomassa. Alla on videolla näkyvä kirjaston käyttöesimerkki laajennettuna ja kommentoituna. Tämä pidempi koodirimpsu lisäksi ympäröi viiva- ja 2D-koodit kuvasta sekä kirjoittaa niiden sisällön tuloskuvaan.

Esimerkissä tarvittavat kuvat ovat pakettina täällä.

```
import cv2
import numpy as np
from os import listdir
from os.path import join
from pyzbar.pyzbar import decode

# Viivakoodikuvat sisältävä kansio
kansio = "viivakoodit"
lista = listdir(kansio)

# Käydään läpi kaikki kansion kuvat ja luetaan niistä viivakoodit
for tiedosto in lista:
    kuvapolku = join(kansio, tiedosto)
    kuva = cv2.imread(kuvapolku)
    info = decode(kuva)

    # Luetaan yksi kerrallaan jokaista koodia vastaava tieto
    for obj in info:
        (x, y, leveys, korkeus) = obj.rect # Koodia rajaava suorakulmio
        pisteet = obj.polygon # Koodin tarkempi reunus
        pts = np.int0(pisteet) # Tyyppimuunnos kokonaisluvuiksi piirtoa varten
        data = obj.data.decode("utf-8") # Koodin lukeminen
        tyyppi = obj.type
        datateksti = "Data: " + str(data)
        tyyppiteksti = "Tyyppi: " + str(tyyppi)

        # Piirretään rajaava suorakulmio punaisella ja rajaava monikulmio (polygoni)
        # vihreällä. Kirjoitetaan luettu koodi ja sen tyyppi kuvaan sinisellä.
        cv2.rectangle(kuva, (x, y), (x + leveys, y + korkeus), (0,0,255),2)
        cv2.polylines(kuva, [pts], True, (0, 255, 0), 3)
        cv2.putText(kuva, datateksti, (x,y-25), cv2.FONT_HERSHEY_SIMPLEX,0.8,(255, 0, 0), 2)
        cv2.putText(kuva, tyyppiteksti, (x,y), cv2.FONT_HERSHEY_SIMPLEX,0.8,(255, 0, 0), 2)

cv2.imshow("tulos", kuva)
cv2.waitKey(0)
```

Harjoitus

Bookmark this page

Kysymys 1: viivakoodikuvien koodien tyypit

1 point possible (ungraded)

Mitä erityyppisiä viivakoodeja (joiden lukeminen onnistuu) viivakoodit-kuvakansion kuvissa on?

- [ ] Code 39
- [ ] Code 128
- [ ] Interleaved 2 of 5
- [ ] EAN13
- [ ] UPC
- [ ] PDF417

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: koodien rajaukset

1 point possible (ungraded)

Kuten huomaat, viivakoodin rajaus ei aina onnistu täydellisesti. Miksi viivakoodin luku silti onnistuu?

- ( ) pelkkä koodin keskialue riittää sen lukemiseen
- ( ) yksi kokonainen sarake koodia riittää sen lukemiseen
- ( ) yksi kokonainen rivi koodia riittää sen lukemiseen
- ( ) rajauksen väri helpottaa koodin lukemista

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: vinot merkkikoodit

1 point possible (ungraded)

Ota kuvia kohteista, joissa on viivakoodeja ja 2D-koodeja. Ota myös kuvia, joissa koodit ovat selkeästi vinossa. Tuleeko hyvin vinojen merkkikoodien kanssa virheitä?

- ( ) sekä viivakoodeissa että 2D-koodeissa
- ( ) vain viivakoodeissa
- ( ) vain 2D-koodeissa
- ( ) ei kummissakaan

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: Tesseractin parameteri minimivarmuus

1 point possible (ungraded)

Kokeile eri lukuja parametriksi `MIN_VARMUUS` ja katso, miten se vaikuttaa tekstin lukemisen lopputulokseen. Millä arvolla kuvan *03-verkkokauppa.jpg* virheellinen niskatyynystä luettu teksti *cy* poistuu? Anna vastauksesi viiden tarkkuudella.

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: Tesseractin ja omat kuvat

1 point possible (ungraded)

Kokeile Tesseractia omiin kuviisi, joissa on tekstiä. Kokeile tarpeen tullen säätää parametria `MIN_VARMUUS`

- ( ) kokeilin
- ( ) en kokeillut

Show answer

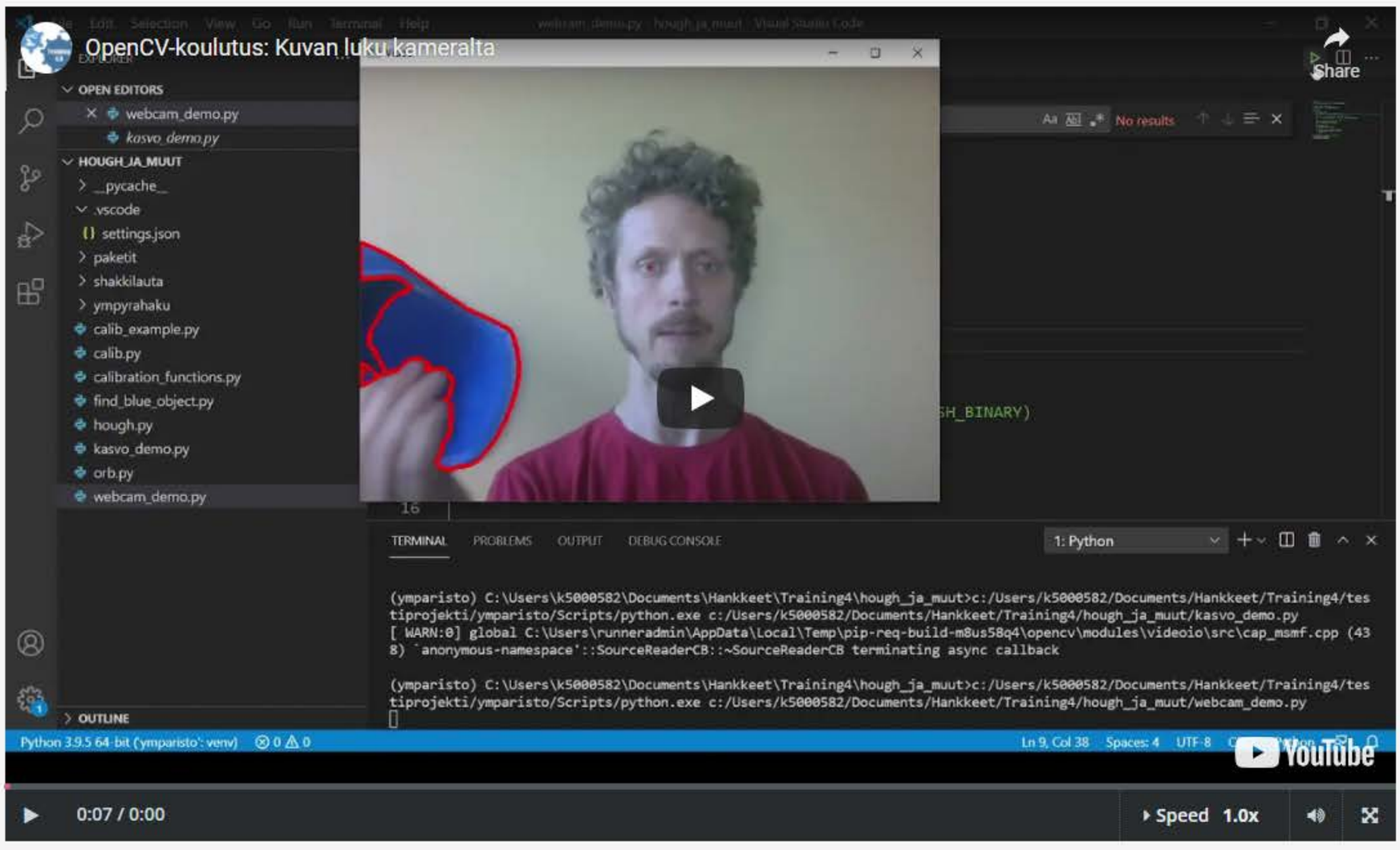
Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kuvanlukuvideo

Bookmark this page

Kuvan lukeminen kameralta



STAFF DEBUG INFO

Kuvan lukeminen kameralta

Bookmark this page

Esimerkkivideossa kuva luettiin tietokoneen kovalevyn sijaan koneen integroidulta web-kameralta. Web-kamera otetaan käyttöön komennolla `webkamera = cv2.VideoCapture(0)` ja kamelta saadaan kuvia komennolla `webkamera.read()`.

Kuvalähde voi olla tietysti mikä tahansa koneeseen liitetty kamera, kunhan sillä on 1) ohjelmointirajapinta (API) sekä 2) Python-kääreluokka. Esimerkiksi Baslerin kameroissa valmistaja tarjoaa mukana Pylon-ohjelmointirajapinnan ja kääreluokkana voi käyttää kolmannen osapuolen tarjoamaa PyPylonia. Niitä hyödynnetään seuraavassa esimerkkivideossa. AVT:n kameroiden kanssa taas ohjelmointirajapintana toimii Vimba, johon on valmistajan sivujen mukaan nykyisin myös Python-rajapinta valmiina. Kolmannen osapuolen tarjoama kääreluokka (jota siis nykyisin ei ilmeisesti tarvita) on Pymba.

Edellisellä videolla nähty värin seuranta (sininen tohveli) on tällä OpenCV:n dokumentaatiosivulla mainitun esimerkin mukainen. Kasvon ja silmien etsinnän koodi taas on alla. Koodi käyttää hyväkseen Haar-luokittimia, jotka tulevat OpenCV:n mukana. Omansa löytyy kasvoille, silmille sekä muutamalle muulle kasvonpiirteelle ja ruumiinosalle. Haar-luokittimet ovat laskennallisesti tehokas tapa etsiä tiettyjä piirteitä (kuten kasvoja) kuvasta käyttämällä hyväksi valon ja varjon eli vaalean ja tumman vaihteluita. Hyvä esimerkkikuvia, jotka auttavat hahmottamaan luokittimen toimintaa sekä pidempi selitys siitä löytyy täältä ja täältä. Esimerkissä etsitään ensin kuvasta kasvot kasvoluokittimella ja sitten silmät löydettyjen kasvojen alueelta silmäluokittimella.

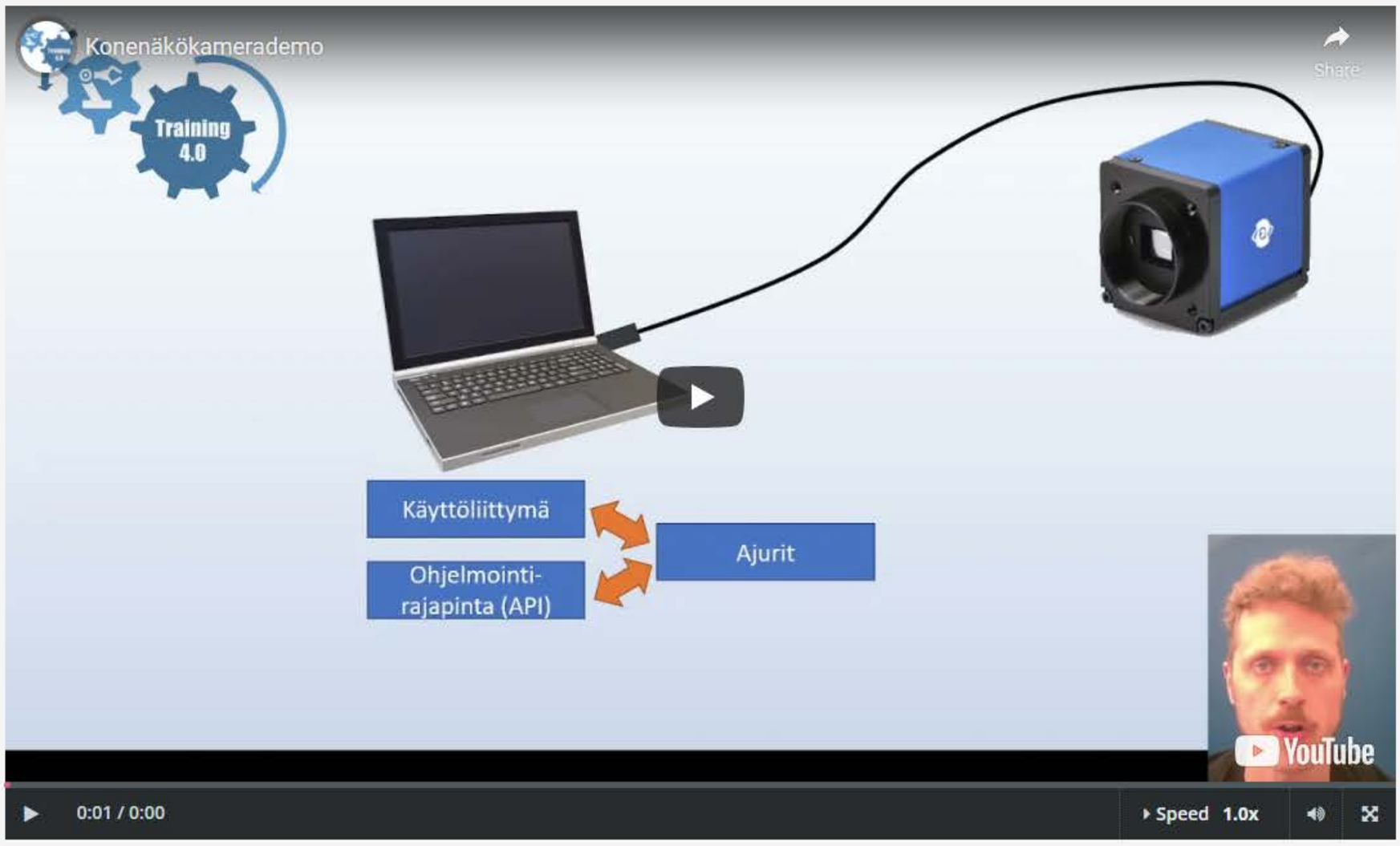
Seuraava video tarjoaa useamman esimerkkisovelluksen, joissa kuvavirtaa luetaan suoraan kameralta. Esimerkeissä käytetään Baslerin konenäkökameraa.

```
import cv2

# Luokittimet kasvoille ja silmille
# VAIHDA NÄMÄ POLUT TARVITTAESSA! NIIDEN TULEE OSOITTAA CV2:N MUKANA TULEVIIN
# HAARCASCADEN XML-TIEDOSTOIHIN
kasvoluokitin = cv2.CascadeClassifier(r'C:\Python\Lib\site-packages\cv2\data\haarcascade_frontalface_default.xml')
silmaluokitin = cv2.CascadeClassifier(r'C:\Python\Lib\site-packages\cv2\data\haarcascade_eye.xml')

# Käynnistetään web-kamera
webkamera = cv2.VideoCapture(0)

# Kuvaus jatkuu, kunnes käyttäjä painaa näppäintä q
while True:
    # Käsitellään web-kameran kuvat yksi kerrallaan
    _, kuva = webkamera.read()
    kuva_hs = cv2.cvtColor(kuva, cv2.COLOR_BGR2GRAY) # Harmaasävymuunnos

    # Etsitään aluksi kasvoja kuvista. Jos kasvoja löytyy, kasvojen sijainti
    # palautetaan muodossa (x, y, leveys, korkeus). Silmien etsintä
    # tehdään vain tästä alueesta, koska silmät ovat kasvoissa.
    kasvot = kasvoluokitin.detectMultiScale(kuva_hs, 1.3, 5)
    for (x, y, lev, kork) in kasvot:
        cv2.rectangle(kuva, (x, y), (x+lev, y+kork), (255, 0, 0), 2)
        roi_hs = kuva_hs[y: y + kork, x: x + lev]
        roi_bgr = kuva[y:y+kork, x:x+lev]
        silmat = silmaluokitin.detectMultiScale(roi_hs)
        for (x, y, lev, kork) in silmat:
            cv2.rectangle(roi_bgr, (x, y), (x + lev, y + kork), (0, 255, 0), 2)

    # Näytetään suorakulmiot kuvan päällä
    cv2.imshow("Video", kuva)

    # Lopetetaan, kun käyttäjä painaa näppäintä 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Kun kaikki on valmista, vapautetaan kamera
video_capture.release()
cv2.destroyAllWindows()
```

Konenäkökameravideo

Bookmark this page

Konenäkökameran käyttö



STAFF DEBUG INFO

Harjoitus

Bookmark this page

Kysymys 1: yhteensopivat kamerat

1 point possible (ungraded)

Mitkä ovat vaatimukset online-toimenpiteissä kuvanlähteenä käytettävälle kameralle?

- [ ] kamera on liitetty tietokoneeseen, jolla ohjelmakoodia ajetaan
- [ ] kemerassa on GiGE-liitäntä
- [ ] kamera on web-kamera
- [ ] kamera on värikamera
- [ ] kameran kenno on korkeintaan 2 MPix
- [ ] koneelle on asennettu kameran ajurit
- [ ] koneelle on asennettu kameran ohjelmointirajapinta (API)
- [ ] koneelle on asennettu kameran firmware
- [ ] jos käytetään Python-ohjelmonitikieltä, koneelle on asennettu myös ohjelmointirajapinnan Python-kääreluokka

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 2: HAAR-luokittimet

1 point possible (ungraded)

Mihin seuraavista OpenCV tarjoaa valmiit HAAR-luokittimet?

- [ ] ihmisen kasvot
- [ ] kissan kasvot
- [ ] koiran kasvot
- [ ] silmä
- [ ] nenä
- [ ] suu
- [ ] leuka
- [ ] vartalo
- [ ] ylävartalo
- [ ] alavartalo
- [ ] käsi
- [ ] jalka

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 3: hymyn tunnistus

1 point possible (ungraded)

Muuta kasvot ja silmät etsivää esimerkkikoodia siten, että se löytää myös hymyn ja ympäröi sen punaisella suorakulmiolla. Kirjoita tässä tarvittavan xml-tiedoston nimi (siis pelkkä tiedostonimi, ei koko polkua).

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 4: värin seuranta

2 points possible (ungraded)

Muuta OpenCV:n dokumentaatiosivujen esimerkki värin seurannasta seuraamaan vihreää väriä. Mitkä tällöin ovat alaraja ja yläraja? Kirjoita alaraja ensimmäiseen kenttään ja yläraja toiseen.

Show answer

Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Kysymys 5: kuvausnopeus

1 point possible (ungraded)

Jos käytössä on 1 GiGE-liitäntä ja 1 MPix värikamera, mikä on maksimikuvausnopeus (montako kuvaa kone voi maksimissaan vastaanottaa sekunnissa)? Anna vastauksesi yhden tarkkuudella.

Show answer

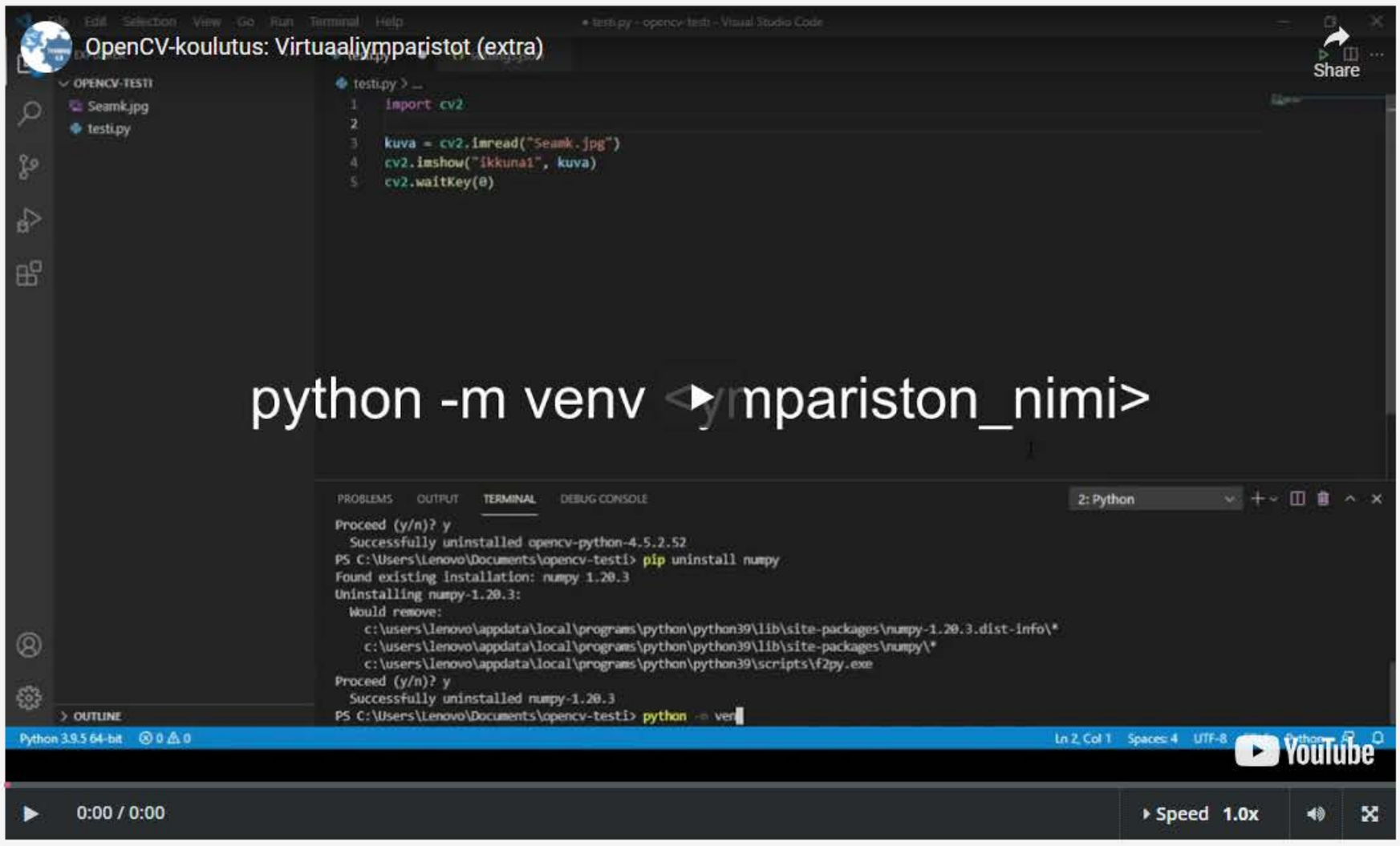
Submit

SUBMISSION HISTORY STAFF DEBUG INFO

Virtuaaliympäristövideo

Bookmark this page

Virtuaaliympäristöt



STAFF DEBUG INFO