Versions historie af video behandling

Gennemprojektet har der været flere forskellige tilgange til at prøve at gå fra billede til kurve. De fleste har været uden succes, men en enkelt virkede i begrænset omfang. Desværre var det omfang så begrænset at det ikke gjorde en forskel for hvor brugbart programmet var.

iteration 1, `Otte.py` og `Greenscreen.py`

For `Greenscreen.py` var konceptet simpelt, isoler nudlen ved at fjerne alle pixels der ikke var orange nok. Programmet `Greenscreen.py` er skrevet i python og bruger bibliotekerne `CV2` og `numpy`. Programmet kører følgende funktioner: - `cv2.VideoCapture()` for at hente et billede fra kameraet. - `cv2.inRange` laver et billede hvor hver pixel der er "mellem" to farver er hvide og alle pixels der ligger uden for farve rækkevidden er sorte. - `remove_isolated_pixels()` for at fjerne små klumper af pixels fra billedet. - `cv2.boundingRect()` som finder koordinaterne til den firkant der kan laves om billedet udfra det mest extreme x og y værdier.

Den anden fil `Otte.py` bruger samme biblioteker og har ikke yderligere funktionalitet en `Greenscreen.py`, dog var det der koordinat placering først blev implementeret og flettet ind i `Greenscreen.py`.

Programmet fungerer ved at en ottendedel af billed-inputtet og var et proof-of-concept til senere idéer.

Samfletningen blev implenteret ved at bruge metoden med at opdele billedet på et subbillede defineret af den kasse `cv2.boundingRect()` har genereret.

iteration 2, `Greenscreen.py` og `Orangescreen.py`

I andet iteration blev koordinat placeringen forbedret, `Orangescreen.py` virkede som testgrund for de nye ændringer koordinat placering blev forbedret så vidt vi kunne se. Vi lavede flere test hvor vi visuelt inspicerede resultatet fra programmet, dog fangede vi ikke mange af de mangler der senere blev fundet i programmet. De fejl der blev fundet var, at der ikke var høj nok præcisition og at nudlen ikke kunne blive vendt på højkant.

iteration 3, `Rotate.py`

Her ses første intruduktion af `cv2.minAreaRect()` som laver den firkant med mindst muligt areal der kan omslutte et objekt. Jeg havde overset en vigtig detalje det jeg læste dokumentationen eller misforstået den betydning, da det senere har været skyld i meget spild arbejde. Når `cv2.minAreaRect()` giver en liste af data er udfaldsrummet begrænset til 0 til -90 grader. Der giver matematisk mening men reduktionen af original dataet har mange afledte effekter på hvad man kan regne.

Programmet i store træk: - Samme som `Greenscreen.py` optil og inkluderende `remove_isolated_pixels()` - Derefter kører `cv2.findContours()` som laver en liste af konturer om det billeder som er angivet. - Konturerne reduceres til ekstreme konturer: `cv2.convexHull()` - Baseret på de ekstreme punkter finder `cv2.minAreaRect()` en firkant.

iteration 4-5, `Rotate.py`

`Rotate.py` kunne køre men virkede ikke hensigtsmæssigt, for at fikse problemet blev der tilført nogle filtrer som gjorde det nemmere for kontur funktionen at finde kanter. Sobel filteret er en "Edge dectection" filter som gør kanter mere tydelige.

```
sx = cv2.Sobel(absolute,cv2.CV_32F,1,0)
sy = cv2.Sobel(absolute,cv2.CV_32F,0,1)
```

Sobel filteret virkede men var ikke nok. det blev løst ved at få pixels som ikke var forbundet til at hænge sammen. Er der 5 pixels horisontalt på række forbinder den følgende kode dem med andre pixels op til 30 pixels væk.

```
rect_kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 30))
threshed = cv2.morphologyEx(absolute, cv2.MORPH_CLOSE, rect_kernel)
```

iteration 6-7, `Rotate.py` og `Rotate_live.py`

`Rotate.py` får en ny funktion `crop_minAreaRect()` og lavet til kode der kan køre med video i `Rotate_live.py`

iteration 8, `Rotate.py`

En punkt rotations funktion bliver implementeret, koden var baseret på et online indslag og har flere fejl.

iteration 9-10, ændringer på mange filer, "Stor ændring"

Tidligere kodes funktioner bliver flyttet til sin egen fil (`Funk.py`) som kan importeres, det gør det lettere at læse koden. Samt nye funktioner i filen:

- `cv2tk(img)` Konverterer fra cv2 læseligt format til tKinter læseligt format
- `rotate_point(point,angle,center)` Roterer et punkt om et andet
- `remove_isolated_pixels(image)` Fjerner isolerede pixels i et billede
- `crop_minAreaRect(img, rect)` Beskær et billede til et roteret rektangel
- `findContour(img)` Finder konturer til et billede

`Setup.py` bliver lavet med en brugerflade til at instruere hvilke værdier `cv2.inRange` skal bruge. `Simple_rotate.py` og `test.py` bliver brugt som sandkasse til at finde ud af hvorfor rotationsfunktionen ikke virker og fikse problemerne.

iteration 11-12, den sidste kamp

Den sidste strækning inden vi måtte opgive vores tilgang. For at programmet skulle virke havde vi brug for at kunne lave punkt placering forskelligt alt efter forholdet mellem siderne på `cv2.minAreaRect()`, dog var det ikke trivielt.

Det sidste forsøg

Sidernes størrelse sammenlignes og billedet roteres så billedet beskæres på den længste led.

```
rotate_compensate = 0

if height > width:
    cropped = funk.rotate_image(cropped, 90)
    rotate_compensate = 90
```

Problemet med denne tilgang er at der ikke er rækkefølge på siderne så vi roterer billedet baseret på dårlige antagelser, variablet `width` er ikke altid bredden af billedet og `height` er ikke nødvendigvist højden.

Her stødte vi også ind i et andet problem, CV2 roterer ikke billeder så de forbliver intakte, kanterne forsvinder.

Dog var dette problem trivielt løst ved at tilføje en funktion, men det løste ikke kodens mere grundlæggende problemer, som blev en del af koden i iteration 3, at punkter bliver sat i en rækkefølge der ikke hænger sammen med konstruktionen af kassen men afstanden til bunden af billedet.

Nye horisonter

Efter vores primære metode gik fra hinanden kom vi på nye idéer, en simplere interaktion der stadig havde dele af vores originale idé. Vi ville lave synthesizeren uden waveform funktionalitet og derfor video behandling uden punktsætning.

Eftersom "nudel-detektion" blev opnået og forfinet gennem forsøget på at lave punktsætningen kunne koden hurtigt bruges til det nye mål.

iteration 2.0, `basic.py`

Koden fra forrige iterationer bliver brugt til at skrive et program der kan finde poolnudlen og dets center koordinat og rotation. Variablerne `h`, `w` og `r` gør det nemt at tilgå koordinat og rotation så koden blot skal skrives sammen med synthesizer delen