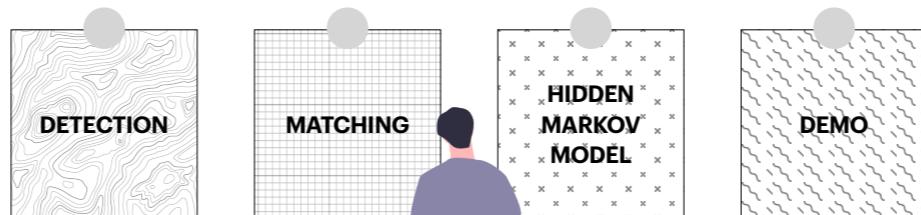




Jochen

CONTENT

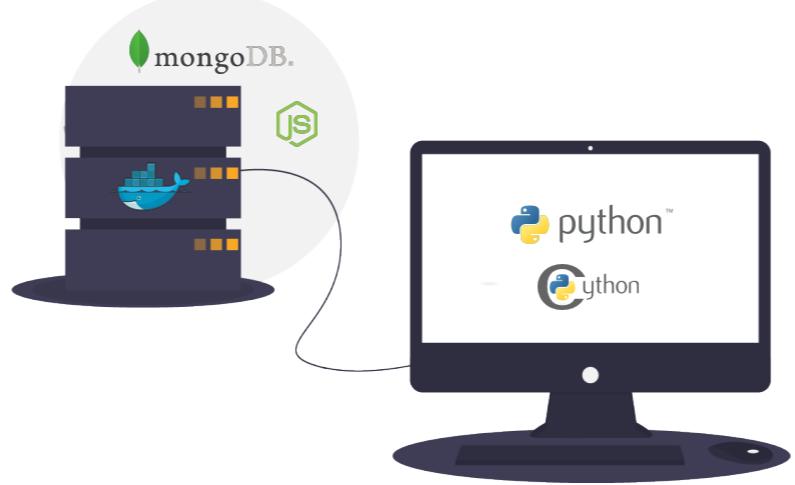


02/06/2020

2

Jochen

SETUP



02/06/2020

3

Jochen

Databank:

- Draait in Docker
- MongoDB
- NodeJS die datadump importeert bij start van de Docker containers

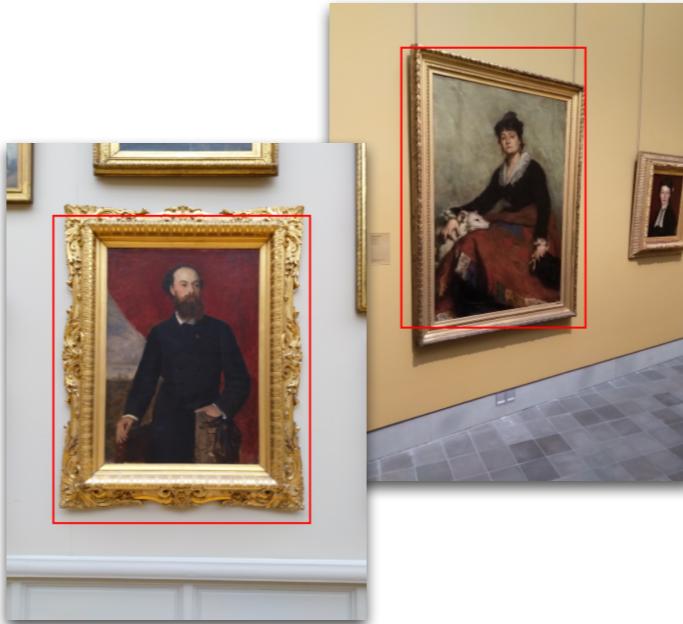
Code: Python + Cython voor computational intensieve onderdelen

DETECTION

Semi-supervised algorithm

02/06/2020

4



Jochen

Algoritme:

Doel: zoveel mogelijk details kwijt geraken + zo snel mogelijk databank te vullen met painting-informatie

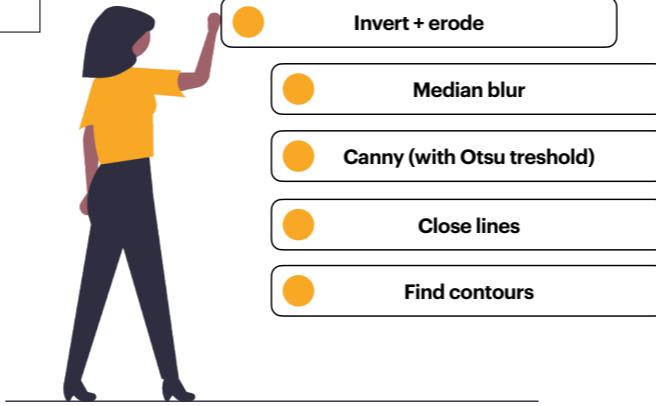
- verkleinen - vergroten
- naar grijswaarden
- dilate (kernel 21x21)
- erode (kernel 21x21)
- median blur
- Canny
- dilate (kernel 9x9)
- contours zoeken
- enkel contours met ratio 1:10 overhouden (horizontaal & verticaal)

Problemen

- Contours zijn niet accuraat genoeg → elk schilderij heeft aanpassingen nodig
- Soms kader wel inbegrepen, soms niet
- Geen rekening met schilderijen onder een hoek bekeken

DETECTION

Unsupervised algorithm



02/06/2020

5

Gillis

1 mean shift segmentation filtering

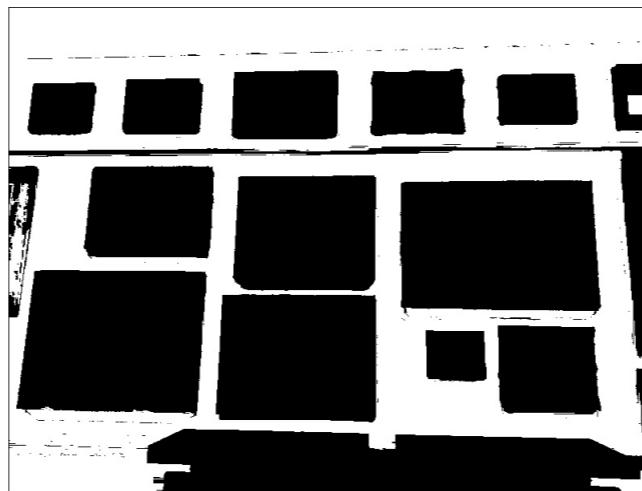


02/06/2020

6

Gillis

2 flood fill

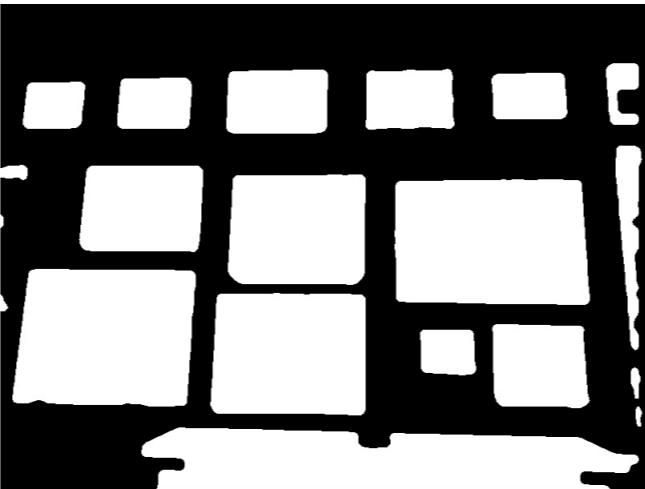


02/06/2020

7

Gillis

3 invert + erode



02/06/2020

8

Gillis

4, 5 & 6 median blur, Canny & close lines



02/06/2020

9

Gillis

7 find contours



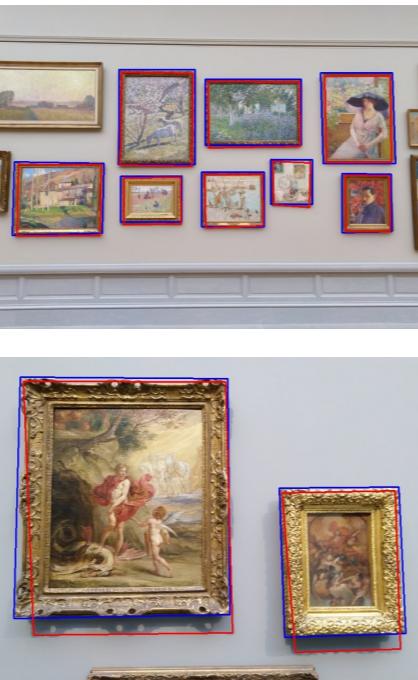
02/06/2020

10

Gillis

SAMPLES

02/06/2020

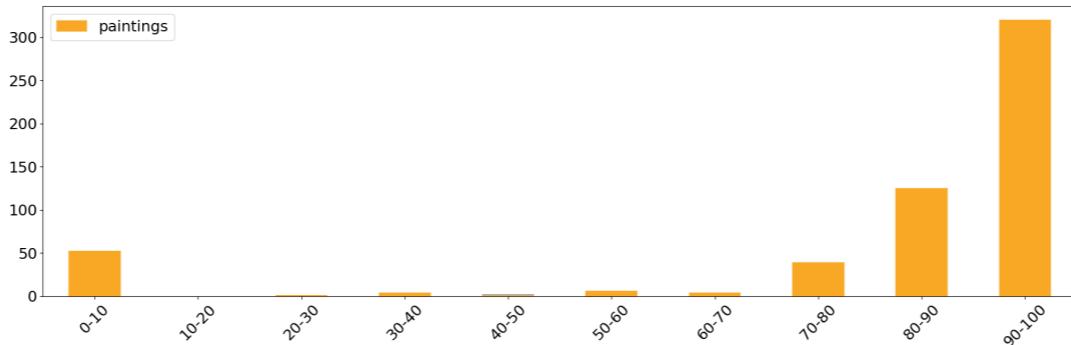


11

Gillis

DETECTION

Detection accuracy ranges for the dataset

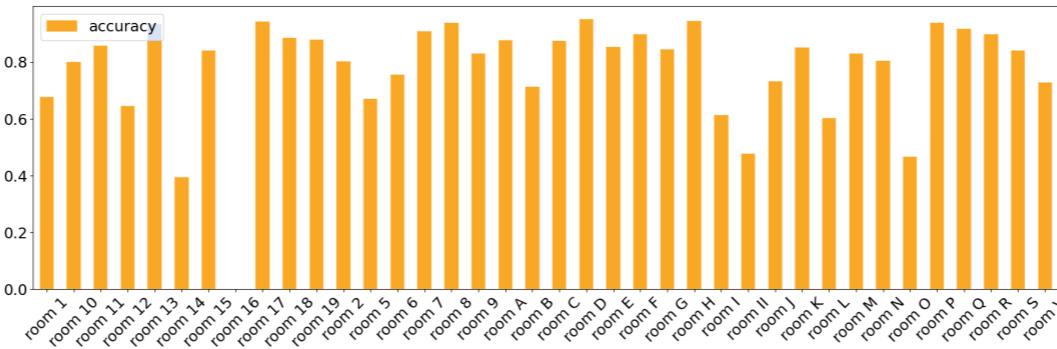


02/06/2020

Gillis

DETECTION

Average room detection accuracy



02/06/2020

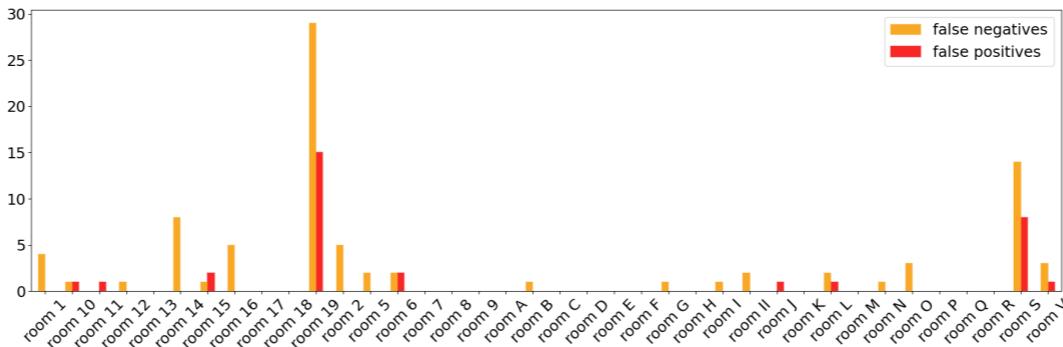
13

Gillis

Average bounding box accuracy: 81,30%

DETECTION

False negatives and false positives per room



02/06/2020

14

Gillis

MATCHING

IDEAS

Histograms
(block & full image)
Local Binary Patterns

ORB
Line segments



02/06/2020

15

Pieter-Jan

Eerste ideeën:

Na het detecteren van de poligonen moeten we deze matchen

Bij Ons eerste idee maakten gebruik van ORB features en line segments, bij deze line segments gingen we textuur detecteren en op basis van de verschillen gaan determineren welk schilderij bekeken wordt.

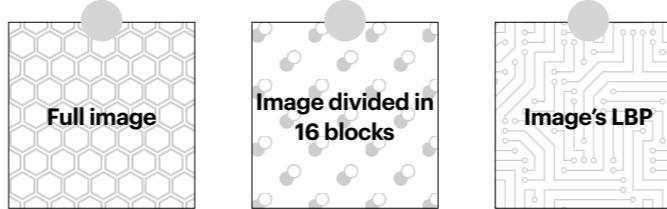
Na het testen hebben we vastgesteld dat deze te traag zijn om realtime schilderijen gaan detecteren. Ze werkten ook niet naar behoren en de opslag in de databank was ook veel te groot.

Daarom zijn we bij het 2de idee overgestapt op het gebruik van histogrammen, deze blijken wel relatief snel!

MATCHING

FINAL CHOICE

Based on histograms of:



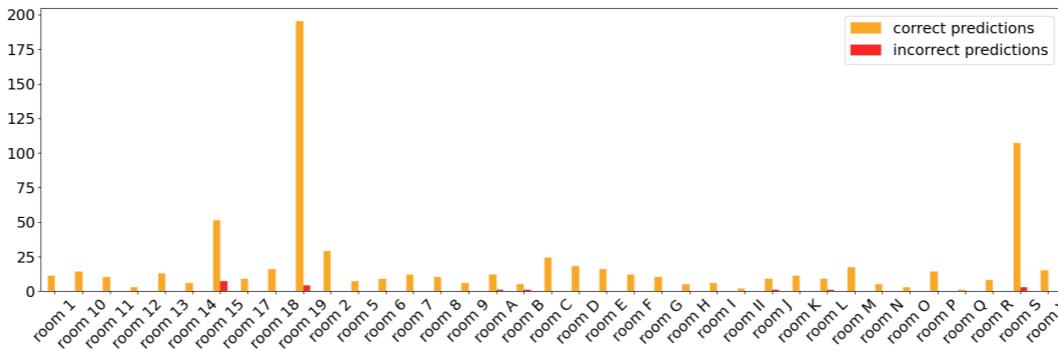
Pieter-Jan

Bij de De uiteindelijke implementatie gebruiken we histogrammen van de volledige afbeelding samen met de histogrammen van de afbeelding verdeeld in 16 blokken en van het Local Binary Pattern van de afbeelding.

Het verschil tussen de histogrammen van een gedetecteerd schilderij en alle gekende schilderijen wordt gecombineerd met aparte gewichten per type histogram. Dit geeft een kans per schilderij waarvan enkel deze boven een threshold overgehouden worden.

MATCHING

Correct and incorrect painting predictions per room



02/06/2020

17

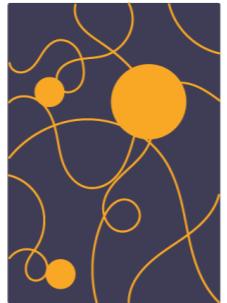
Pieter-Jan

In deze grafiek valt op te merken dat zeer weinig foutieve voorspellingen worden gemaakt in de gekregen dataset. Het valt ook op dat kamer 15, 19 en S enorm boven de rest uitsteken, dit komt omdat de foto's uit deze kamers veel schilderijen bevatten. Deze kamers beïnvloeden vaak de voorspellingen tijdens de video, meer hierover tijdens de demo.

Resultaten:

- 835/836 schilderijen (uit db) werden uniek gematched met een average matching probability van 92,24%
- In combinatie met detectie-algoritme: 702/762 schilderijen correct gematched
- Testset: 158/338 schilderijen correct gematched

HIDDEN MARKOV MODEL



["19" , 0.456]
["3" , 0.061]
["19" , 0.320]
["J" , 0.106]
["19" , 0.764]



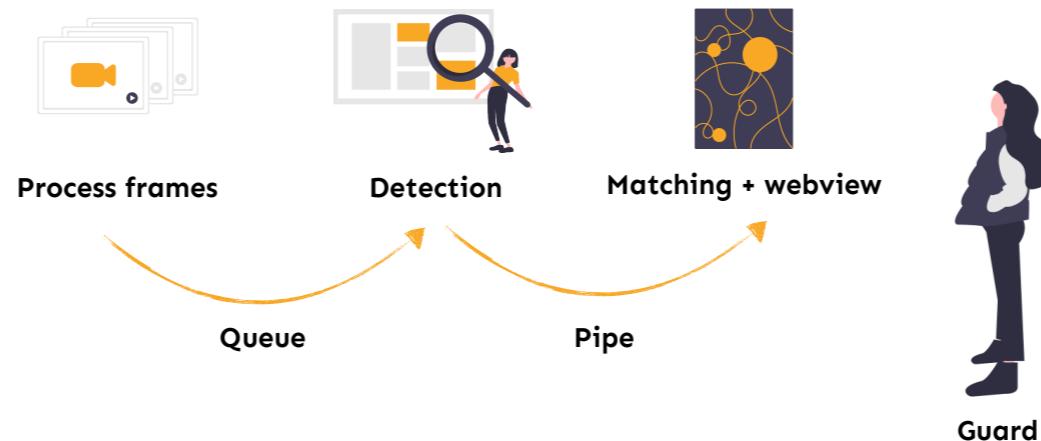
02/06/2020

18

Andreas

Uit de matching verkrijgen we een hoop kansen die een bepaald schilderij linken aan een kamer. Wij hebben een alternatief Hidden Markov Model uitgedokterd met een geschiedenis van waarnemingen. Dit model aggregert de verkregen kansen per kamer om vervolgens o.b.v. de mogelijke kamers een voorspelling te maken. De mogelijke kamers in ons model bestaan uit de huidige kamer en zijn directe buren. Uiteindelijk wordt de kamer met de hoogste kans als voorspelling genomen. Indien deze voorspelling mogelijk is en dus geen teleportatie veroorzaakt, wordt de voorspelling aan de geschiedenis toegevoegd. Indien de geschiedenis groot genoeg is, wordt de meest voorkomende kamer genomen als voorspelling. Deze geschiedenis is een circulaire buffer van een vaste grootte.

DEMO



02/06/2020

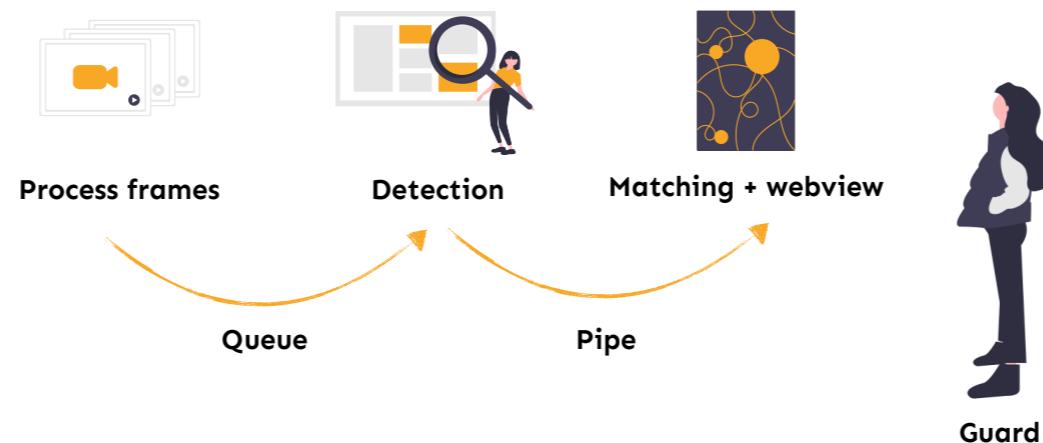
19

Thomas

Alvorens over te gaan tot de effectieve demo, eerst kort uitleggen hoe onze oplossing ontwikkeld is met performantie in het achterhoofd. De grootste vertraging komt door de mean shift segmentation filter en werd opgelost door elk frame met een factor 5 te verkleinen voor de detectie. Nadien worden de resulterende punten voor elk gedetecteerd schilderij geschaald naar het frame van normale grootte.

Een tweede performantieverbetering werd bekomen door 3 kindprocessen te maken met elke een specifiek doel. Doel uitleggen!

DEMO



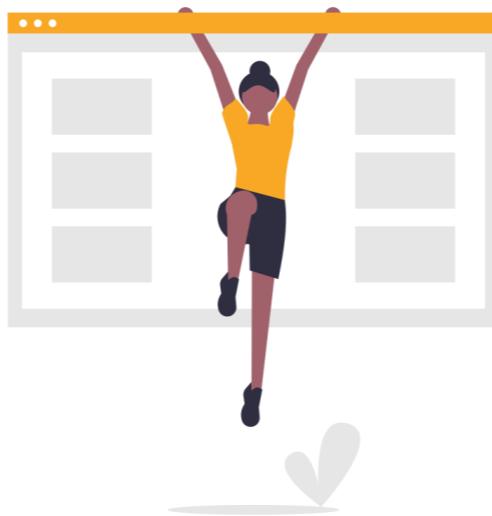
02/06/2020

20

Thomas

Stel één proces faalt, dan zal de guard dit merken en alle andere resources en processen afsluiten. Daarna sluit het zichzelf af.

DEMO



02/06/2020

21

Thomas

DEMO TIME!

