
Is this your card?
The Magic of Computer Vision

IANADS

- @guyroyse (probably)

@cr0wst

nexmo® | The Vonage® API Platform

IANAPD

@cr0wst

nexmo® | The Vonage® API Platform


EXTRA JOKER
JOKER
JOKER
@cr0wst
nexmo
The Vonage API Platform

"Getting someone to pick a card is almost impossible. Go to a party, fan out a deck of cards, and ask someone to pick a card. Everyone will say no."

–Penn Jillette


EXTRA JOKER
JOKER
JOKER
@cr0wst
nexmo
The Vonage API Platform

Every magic trick consists of three parts, or acts.

–The Prestige

@cr0wst

nexmo® | The Vonage® API Platform

The first part is called the pledge, the magician shows you something ordinary.

–The Prestige

The second act is called the turn, the magician takes the ordinary something and makes it into something extraordinary.

–The Prestige

...making something disappear isn't enough; you have to bring it back. That's why every magic trick has a third act, the hardest part, the part we call "The Prestige".

–The Prestige

Where is the real magic?

@cr0wst

nexmo® | The Vonage® API Platform

Computer Vision aims to train computers to understand the visual world.

We're going to aim to answer two questions.

@cr0wst

nexmo® | The Vonage® API Platform

Where is the card in the video?

@cr0wst

Which card is it?

@cr0wst

nexmo® | The Vonage® API Platform

```
import cv2
import imutils

def main():
    cap = cv2.VideoCapture(1)
    while True:
        ret, frame = cap.read()
        frame = imutils.resize(frame, 640)
        cv2.imshow("Image", frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

main()
```

@cr0wst

```
import cv2
import imutils

def main():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        frame = imutils.resize(frame, 640)
        cv2.imshow("image", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cv2.destroyAllWindows()

main()
```

Open Computer Vision Library

Convenience functions for basic image processing.

@cr0wst

nexmo® | The Vonage® API Platform

Capture the video

```
cap = cv2.VideoCapture(1)
    while True:
        ret, frame = cap.read()
        frame = imutils.resize(frame, 640)
        cv2.imshow("Image", frame)
```

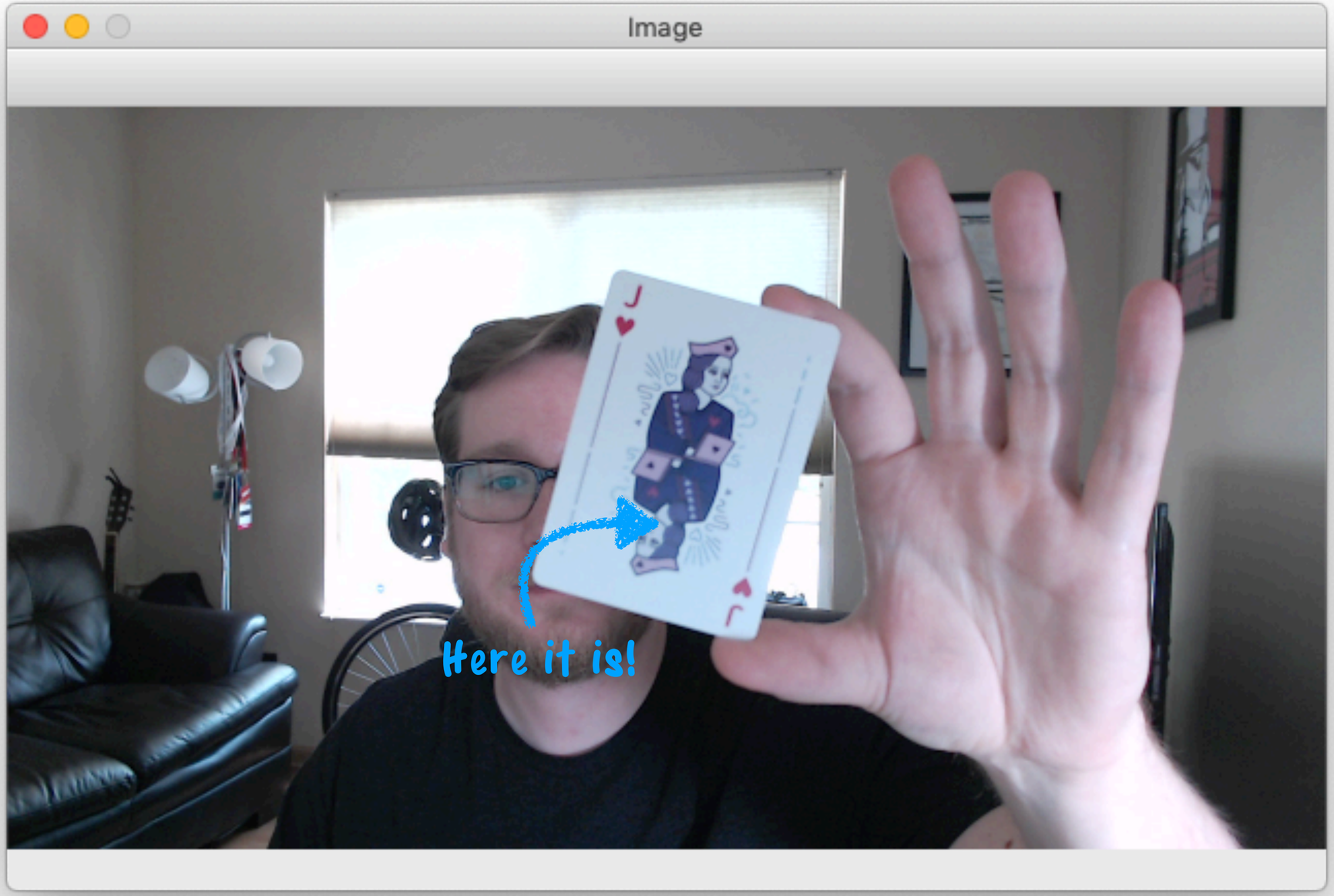
Display each frame as a continuous video loop.

@cr0wst

nexmo® | The Vonage® API Platform

@cr0wst

nexmo® | The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform

Computers need help.

@cr0wst

nexmo® | The Vonage® API Platform

@cr0wst

nexmo® | The Vonage® API Platform

```python
import numpy as np

def normalize(frame):
    frame = imutils.resize(frame, 640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.bilateralFilter(gray, 10, 15, 15)

    edges = cv2.Canny(blur, 50, 150, True)
    kernel = np.ones((5, 5), np.uint8)
    dilate = cv2.dilate(edges, kernel, iterations=1)

    return dilate
```

@cr0wst

```
import numpy as np

def normalize(frame):
    frame = imutils.resize(frame, 640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.bilateralFilter(gray, 10, 15, 15)
    edges = cv2.Canny(blur, 50, 150, True)
    kernel = np.ones((5, 5), np.uint8)
    dilate = cv2.dilate(edges, kernel, iterations=1)
```

Adds arrays, matrices, large number support, and math functions.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

This flag is the type of conversion

@cr0wst

nexmo® The Vonage® API Platform

```
import numpy as np

def normalize(frame):
    frame = imutils.resize(frame, 640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.bilateralFilter(gray, 10, 15, 15)
    edges = cv2.Canny(blur, 50, 150, True)
    kernel = np.ones((5, 5), np.uint8)
    dilate = cv2.dilate(edges, kernel, iterations=1)

    return dilate
```

This is the sigma space

```
blur = cv2.bilateralFilter(gray, 10, 15, 15)
```

This is the size of the filter

@cr0wst

@cr0wst

nexmo® | The Vonage® API Platform

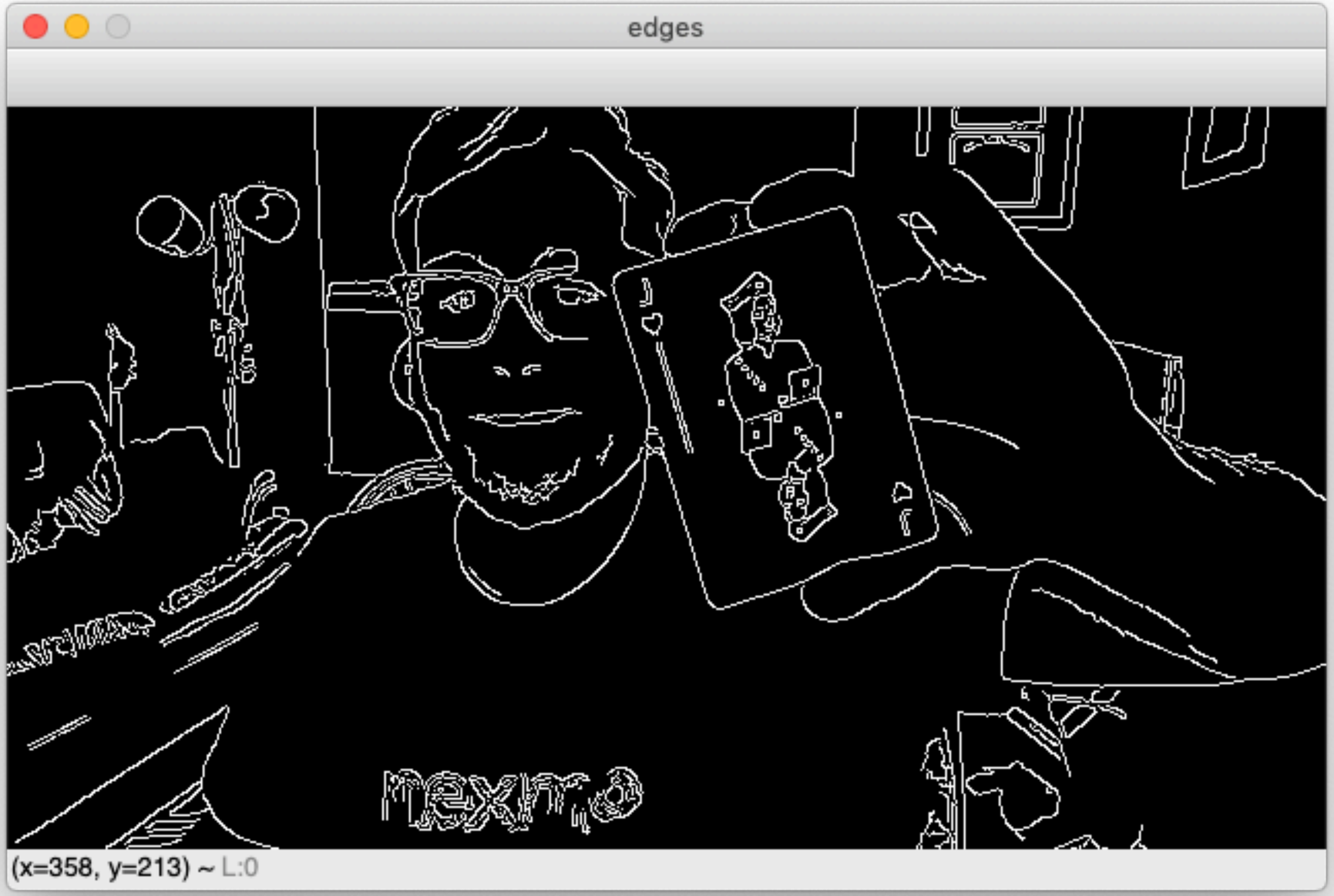
```
edges = cv2.Canny(blur, 50, 150, True)
```

Maximum Threshold

Minimum Threshold

@cr0wst

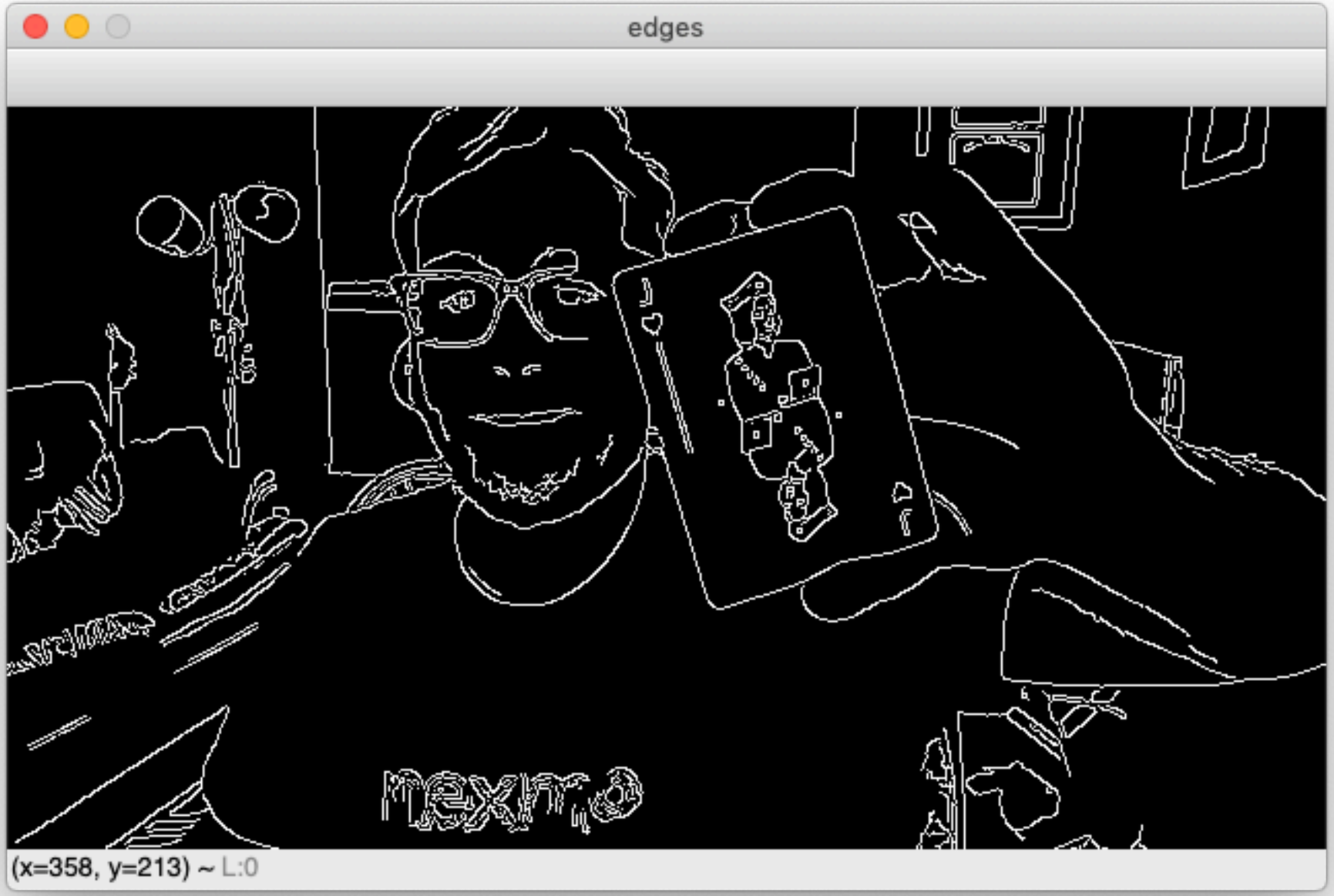
nexmo® | The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform

Why Canny Edge Detection?



@cr0wst

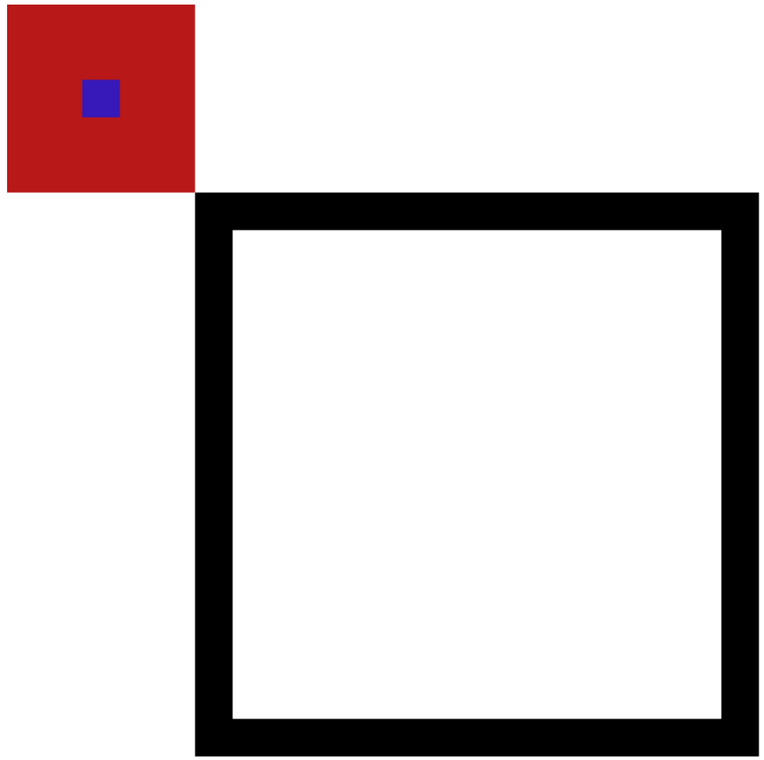
nexmo® | The Vonage® API Platform

```
kernel = np.ones((5, 5), np.uint8)
dilate = cv2.dilate(edges, kernel, iterations=1)
```

How many times to dilate

@cr0wst

nexmo® The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform

@cr0wst

nexmo® The Vonage® API Platform

Cards are kind of rectangular

@cr0wst

nexmo® | The Vonage® API Platform

@cr0wst

nexmo® The Vonage® API Platform

```
def get_contours(dilate):
    contours = cv2.findContours(dilate, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)
    return sorted(contours, key=cv2.contourArea, reverse=True)
```

@cr0wst

nexmo® | The Vonage® API Platform

The approximation algorithm. This has to do with how the points are stored.

```
contours = cv2.findContours(dilate, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

The retrieval mode. This has to do with edge hierarchies.

@cr0wst

nexmo® | The Vonage® API Platform

```
sorted(contours, key=cv2.contourArea, reverse=True)
```

@cr0wst

nexmo® | The Vonage® API Platform

Remember, rectangles are what we want.

@cr0wst

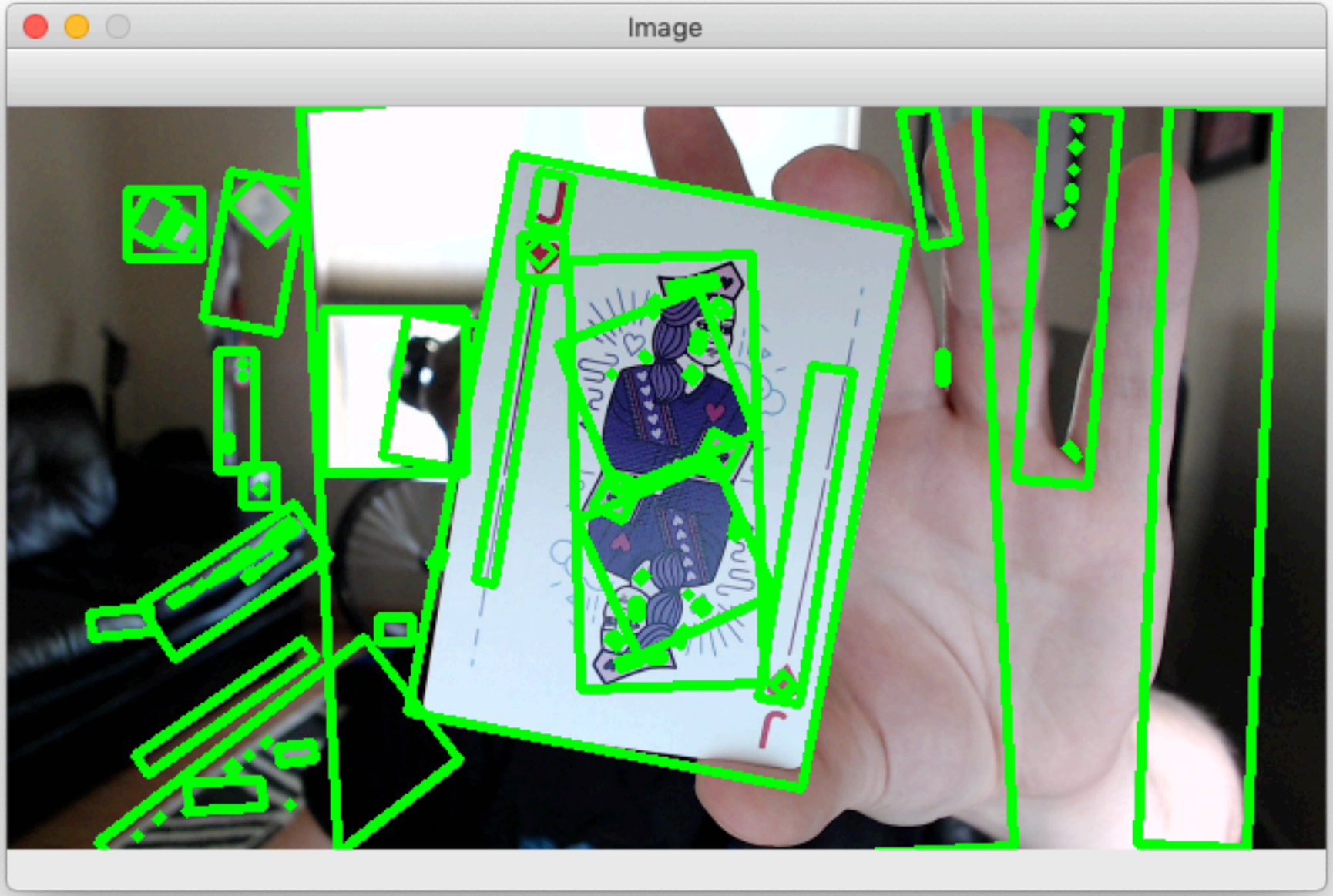
nexmo® | The Vonage® API Platform

```
def process(frame):
    image = frame.copy()
    dilate = normalize(frame)
    contours = get_contours(dilate)

    for c in contours:
        rect = cv2.minAreaRect(c)
        box = cv2.boxPoints(rect)
        box = np.int0(box)

        cv2.drawContours(image, [box], -1, (0, 255, 0), 3)
```

```
def process(frame):
    image = frame.copy()
    dilate = normalize(frame)
    contours = get_contours(dilate)

    for c in contours:
        rect = cv2.minAreaRect(c)
        box = cv2.boxPoints(rect)
        box = np.int0(box)

        cv2.drawContours(image, [box], -1, (0, 255, 0), 3)
```

@cr0wst

nexmo® | The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform

My playing cards have an aspect ratio of around 1.43

@cr0wst

nexmo® | The Vonage® API Platform

```
((x, y), (w, h), a) = rect = cv2.minAreaRect(c)
box = cv2.boxPoints(rect)
box = np.int0(box)

# sometimes w and h are reversed for turned images
ar = w / float(h) if w > h > 0 else h / float(w)
if 1.40 <= ar <= 1.45:
    cv2.drawContours(image, [box], -1, (0, 255, 0), 3)
```

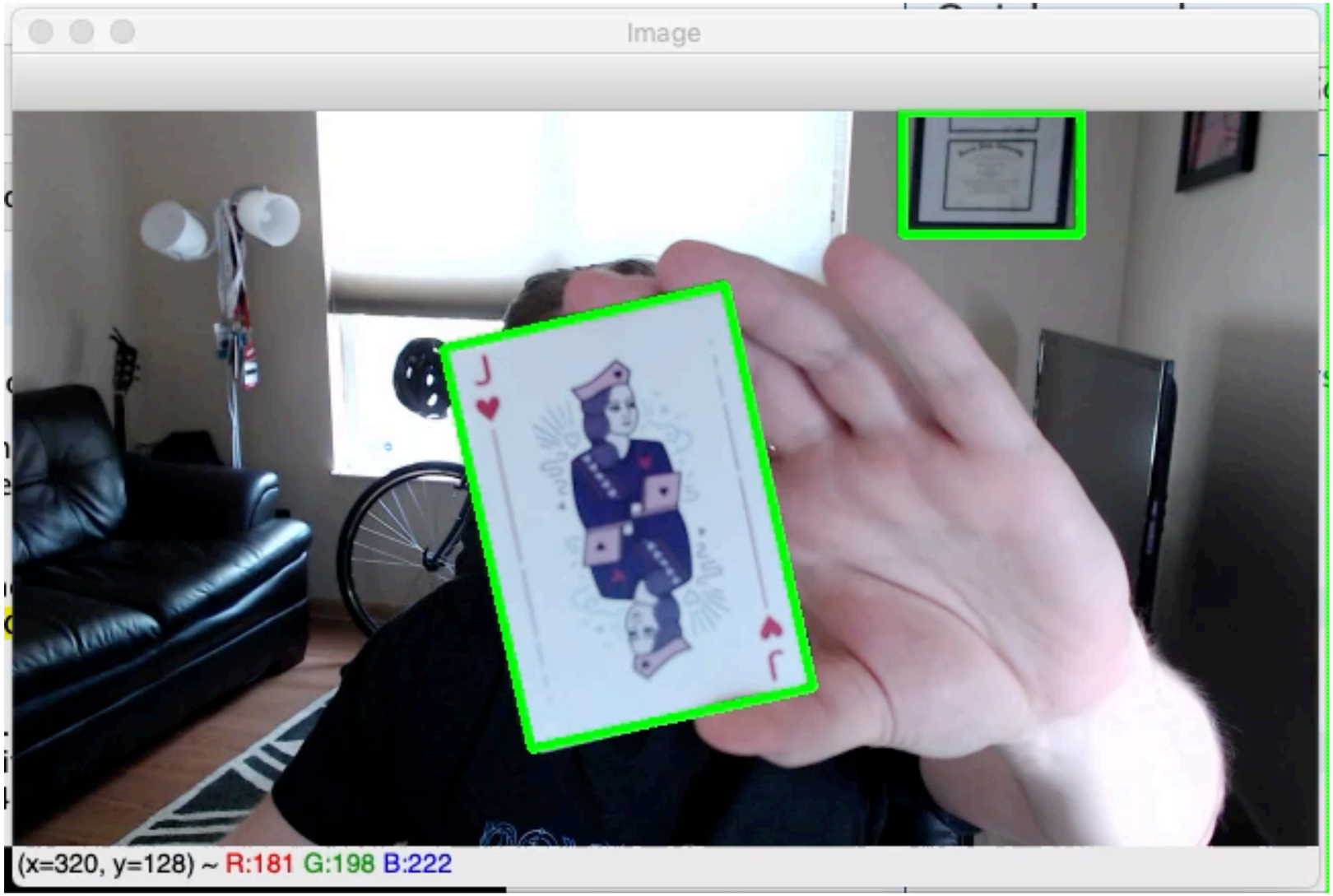
```
((x, y), (w, h), a) = rect = cv2.minAreaRect(c)
```

@cr0wst

nexmo® The Vonage® API Platform

```
# sometimes w and h are reversed for turned images
ar = w / float(h) if w > h > 0 else h / float(w)
if 1.40 <= ar <= 1.45:
    cv2.drawContours(image, [box], -1, (0, 255, 0), 3)
```

@cr0wst



@cr0wst

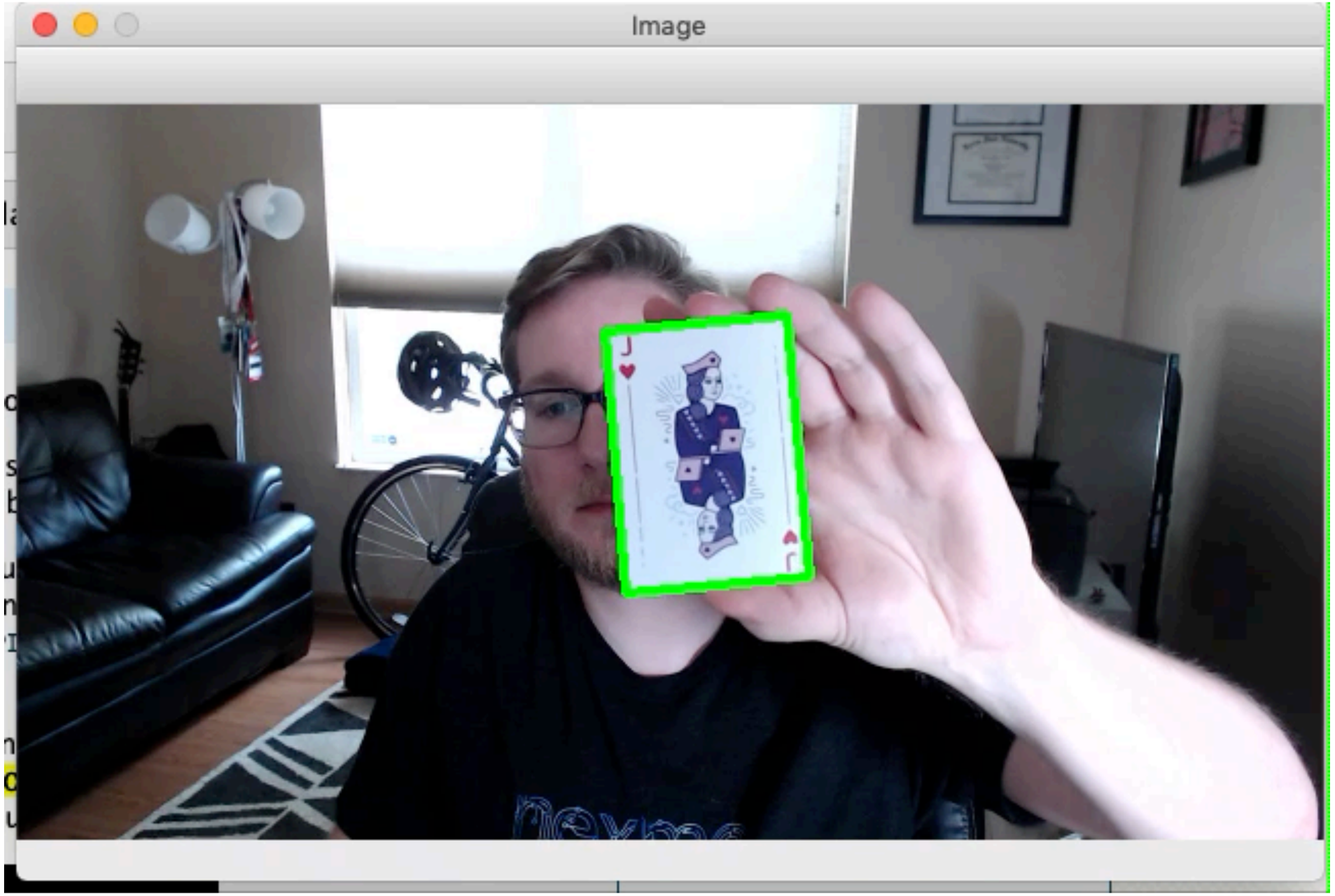
nexmo® The Vonage® API Platform

```
for c in contours[:2]:
    ((x, y), (w, h), a) = rect = cv2.minAreaRect(c)
    box = cv2.boxPoints(rect)
    box = np.int0(box)

    ar = w / float(h) if w > h > 0 else h / float(w)
    if 1.40 <= ar <= 1.45:
        cv2.drawContours(image, [box], -1, (0, 255, 0), 3)

    cv2.imshow("Image", image)
```

```
for c in contours[:2]:
    ((x, y), (w, h), a) = rect = cv2.minAreaRect(c)
    box = cv2.boxPoints(rect)
    box = np.int0(box)
    ar = w / float(h) if w > h > 0 else h / float(w)
    if 1.40 <= ar <= 1.45:
        cv2.drawContours(image, [box], -1, (0, 255, 0), 3)

cv2.imshow("Image", image)
```

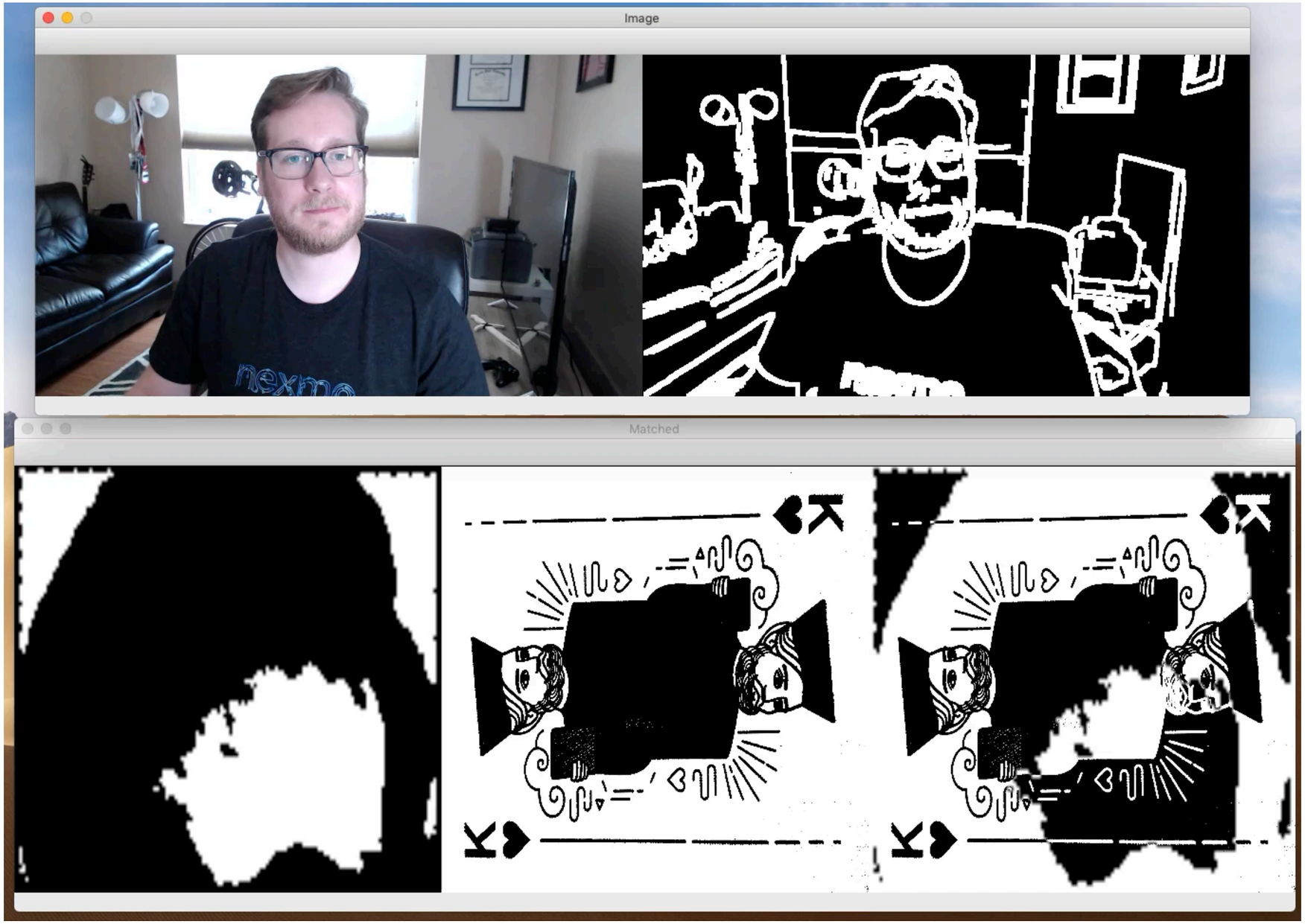
@cr0wst

nexmo® | The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform



@cr0wst

nexmo® | The Vonage® API Platform

Simplify when possible.

@cr0wst

nexmo® | The Vonage® API Platform

Dive in

@cr0wst

nexmo® | The Vonage® API Platform

I'm Steve!

@cr0wst

That's a zero

steve@smcrow.net

https://cr0w.st/cardvision

@cr0wst

nexmo® | The Vonage® API Platform