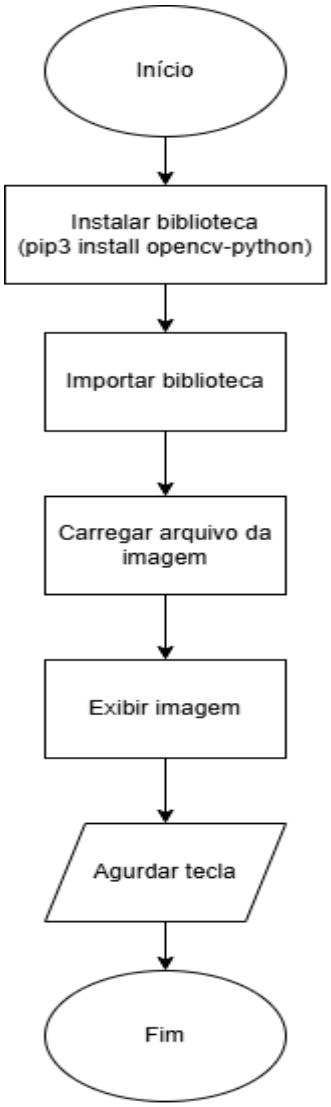
Relatório de progresso: Reconhecimento de Display 7 segmentos com OpenCV

06/06/2025

Visão geral

Instalação da biblioteca e primeiros passos para leitura do arquivo e exibição de imagem em linguagem Python.

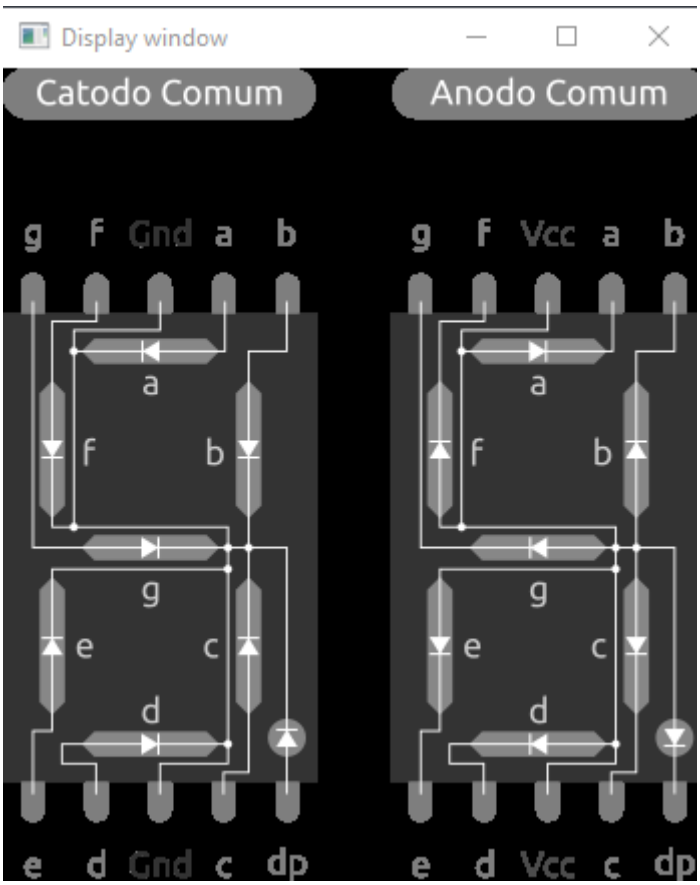
Fluxograma



Código

```python
import cv2 as cv

img = cv.imread("img/display.png", cv.IMREAD_GRAYSCALE)
cv.imshow("Display window", img)
k = cv.waitKey(0)
```

Saída



24/06/2025

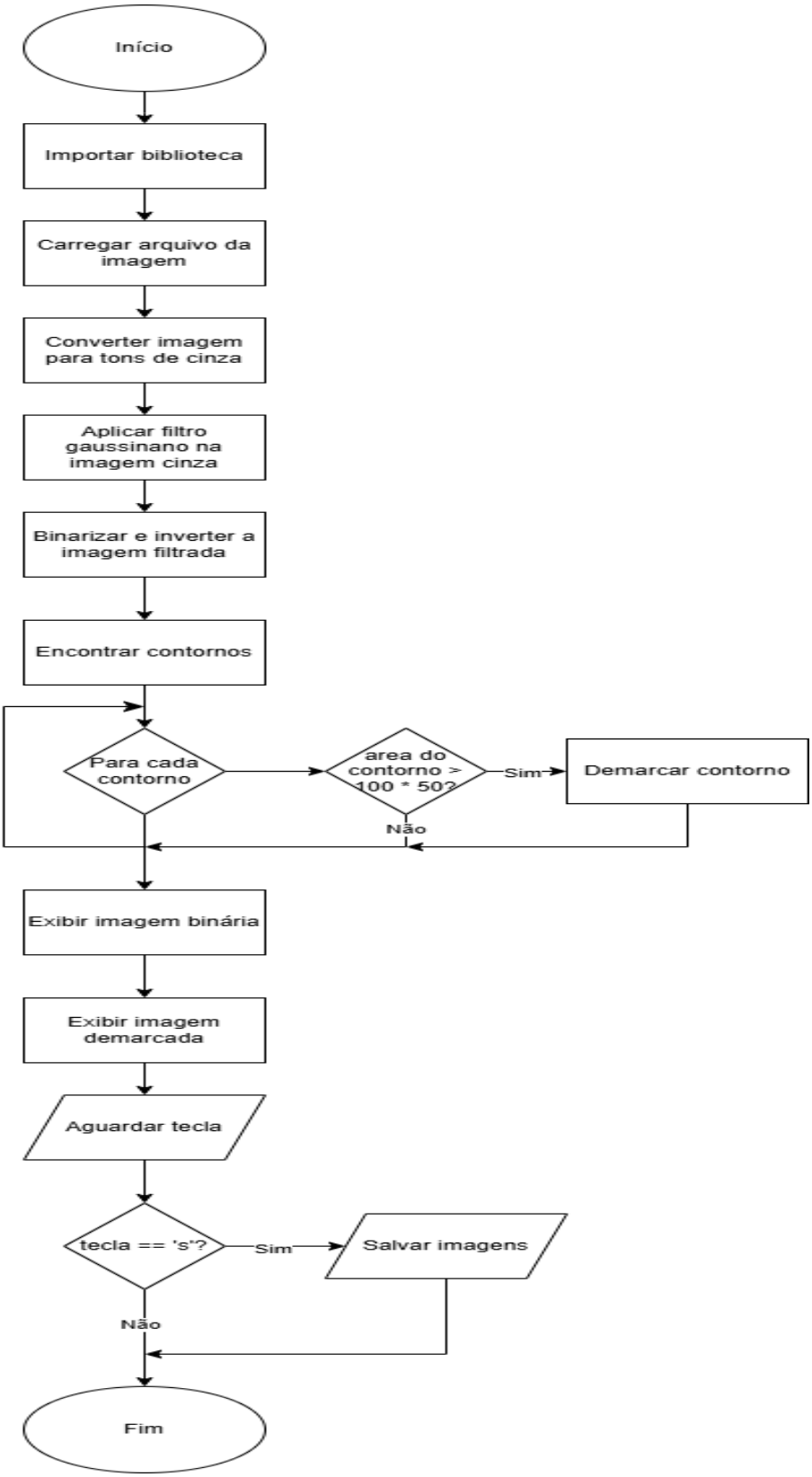
Visão Geral

Testes básicos de aplicação de filtros de cor e binarização da imagem para extração de dados.

Uma imagem amostral de um multímetro foi escolhida para a realização do procedimento. As etapas foram:

- Converter a imagem para tons de cinza
- Aplicar filtro gaussiano para “borrar” levemente a imagem e reduzir ruídos nas bordas
- Binarizar a imagem apenas para tons pixels pretos ou brancos utilizando um valor limiar de intensidade
- Inverter a imagem binária
- Encontrar contornos utilizando os métodos internos do OpenCV
- Filtrar os contornos encontrados (carece de ajuste)
- Demarcar contornos

Fluxograma

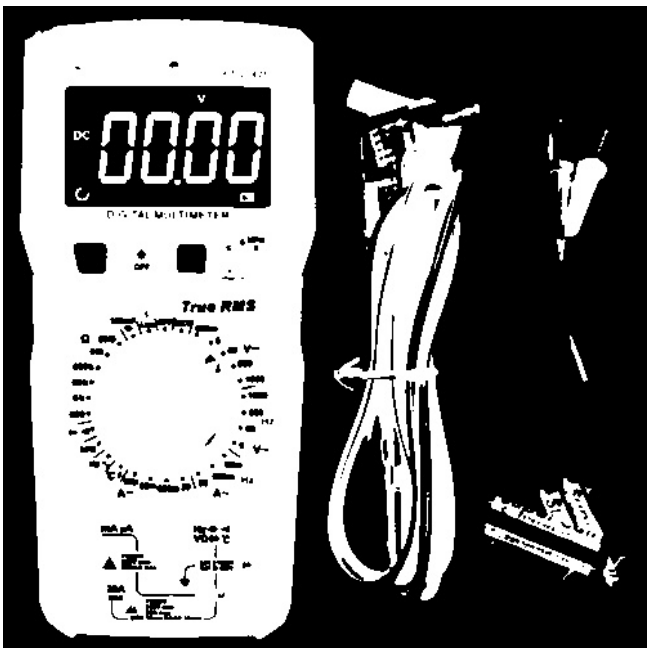

Início
Importar biblioteca
Carregar arquivo da imagem
Converter imagem para tons de cinza
Aplicar filtro gaussinano na imagem cinza
Binarizar e inverter a imagem filtrada
Encontrar contornos
Para cada contorno
area do contorno > 100 * 50?
Sim
Demarcar contorno
Não
Exibir imagem binária
Exibir imagem demarcada
Aguardar tecla
tecla == 's'?
Sim
Salvar imagens
Não
Fim

Código

```python
import cv2 as cv

# Carregar e redimensionar imagem
img_dir = r'.\img'
img = cv.imread(f"{img_dir}\\multimetro.jpg")
img = cv.resize(img, (500, 500))

# Converter imagem para preto e branco e aplicar filtro gaussiano
gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
blurred_img = cv.GaussianBlur(gray_img, (3, 3), 0)

# Binarizar imagem
_, thresh = cv.threshold(blurred_img, 120, 255, cv.THRESH_BINARY_INV)

# Encontrar contornos
contours, _ = cv.findContours(thresh, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

# Filtrar contornos através de uma aréa mínima para evitar pequenos ruídos (carece de ajustes)
for cnt in contours:
  area = cv.contourArea(cnt)
  if area > 100 * 50:
      x, y, w, h = cv.boundingRect(cnt)
      cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv.imshow("Imagem binarizada", thresh)
cv.imshow("Imagem com contorno", img)
k = cv.waitKey(0)

if chr(k) == 's':
  cv.imwrite(f'{img_dir}\\multimetro_binario.jpg', thresh)
  cv.imwrite(f'{img_dir}\\multimetro_demarcado.jpg', img)
```

Entrada

Saída

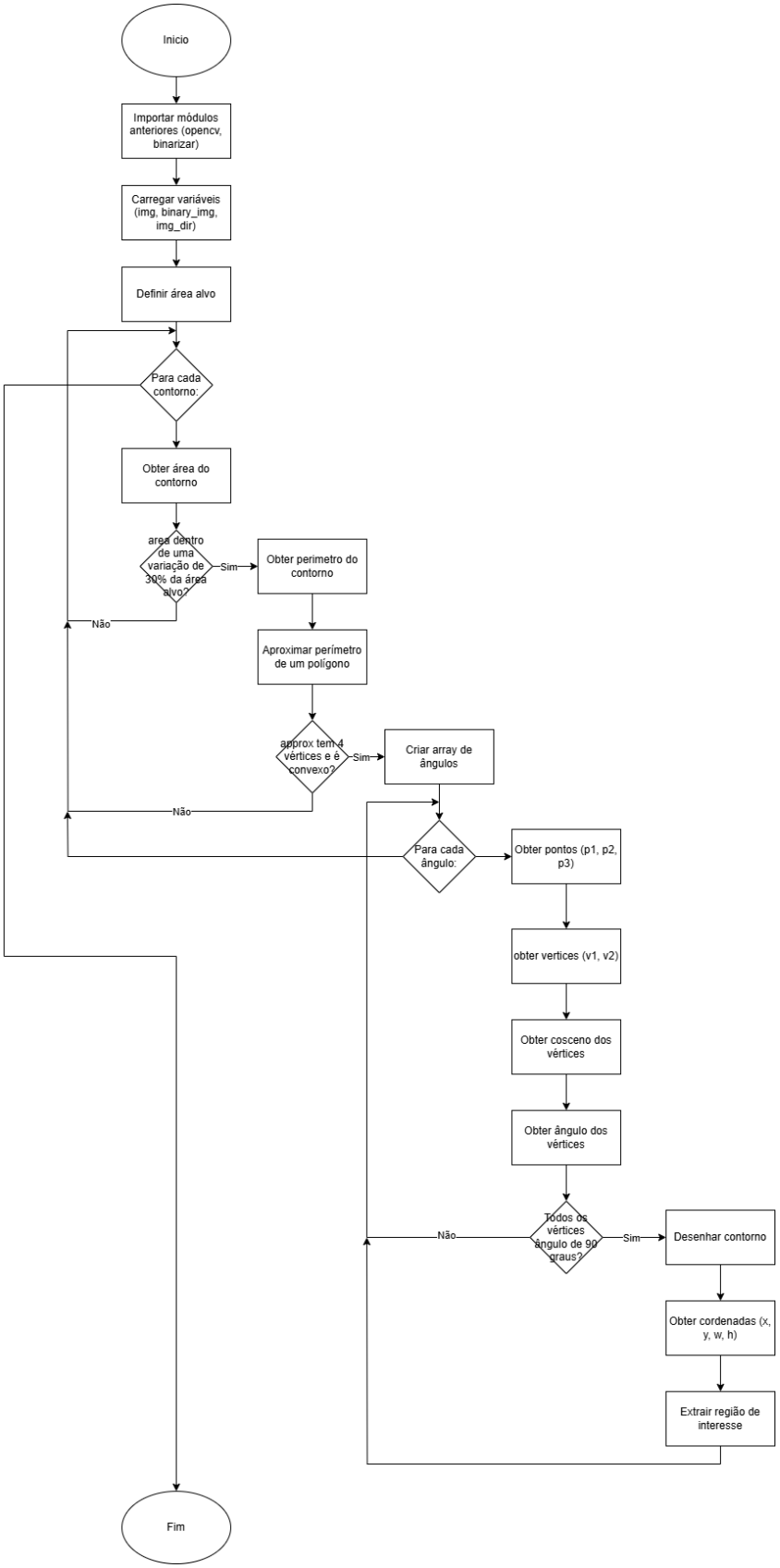


28/06/2025 (1)

Visão geral

Obtenção da região de interesse (ROI) do display através de algoritmos de aproximação em cima do perímetro do contorno e refinamento da área de filtro.

Fluxograma


Inicio
Importar módulos anteriores (opencv, binarizar)
Carregar variáveis (img, binary_img, img_dir)
Definir área alvo
Para cada contorno:
Obter área do contorno
area dentro de uma variação de 30% da área alvo?
Sim
Não
Obter perimetro do contorno
Aproximar perímetro de um polígono
approx tem 4 vértices e é convexo?
Sim
Não
Criar array de ângulos
Para cada ângulo:
Obter pontos (p1, p2, p3)
obter vertices (v1, v2)
Obter cosceno dos vértices
Obter ângulo dos vértices
Todos os vértices ângulo de 90 graus?
Não
Sim
Desenhar contorno
Obter cordenadas (x, y, w, h)
Extrair região de interesse
Fim

Código

```python
import cv2 as cv
import numpy as np
import binarizar

img_dir = binarizar.img_dir
img = binarizar.img
binary_img = binarizar.thresh

# Filtrar contornos através de uma aréa mínima para evitar pequenos ruídos
target_area = 162 * 97 # Número literal. Valores mais adequados precisam ser encontrados.
for cnt in binarizar.contours:
  area = cv.contourArea(cnt)
  # Verifica se a area do contorno está dentro das tolerâncias
  if area > target_area * 0.7 and area < target_area * 1.3:
      # Tenta aproximar o perimetro do contorno a um retângulo
      perimeter = cv.arcLength(cnt, True)
      approx = cv.approxPolyDP(cnt, 0.02 * perimeter, True)

      # Verifica se o contorno aproximado possui
      if len(approx) == 4 and cv.isContourConvex(approx):
        angles = []
        for i in range(4):
          p1 = approx[i][0]
          p2 = approx[(i + 1) % 4][0]
          p3 = approx[(i + 2) % 4][0]

          v1 = p1 - p2
          v2 = p3 - p2
          cos_angle = np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))
          angle = np.degrees(np.arccos(cos_angle))
          angles.append(angle)

          if all(a >= 85 and a <= 95 for a in angles):
            cv.drawContours(img, [cnt], 0, (0, 255, 0), 3)
            x, y, w, h = cv.boundingRect(cnt)
            roi = img[y:y+h, x:x+w]

cv.imshow("Imagem binarizada", binary_img)
cv.imshow("Imagem com contorno", img)
cv.imshow("Display", roi)
k = cv.waitKey(0)

if chr(k) == 's':
  contour_path = f'{img_dir}\\multimetro_demarcado.jpg'
  display_path = f'{img_dir}\\display.jpg'
  cv.imwrite(contour_path, img)
  cv.imwrite(display_path, roi)
  print(f'Imagem guardada em: {contour_path}'.replace('.\\', ''))
  print(f'Imagem guardada em: {display_path}'.replace('.\\', ''))
```

Entrada

Saída

28/06/2025 (2)

Visão geral

Reorganização das rotinas de pré-processamento e tratamento de imagem em funções para reutilização e testes com diversas imagens amostrais. O processo foi reorganizado para isolar o aparelho (um multímetro, neste caso) e depois identificar o display.

Entradas


000
OFF
hFE
BRASFORT
Ref.: 8522
CAT II
COM


FLUKE 115
6000
OFF
COM


ET-2507B
TRMS
V
LOZ
AC
599.9
BL
0 10 20 30 40 50 60
True RMS
6000 Counts
RANGE
MAX MIN
REL
Hz%
HOLD
SELECT
mV
Hz%
V
µA
OFF
mA
LoZ V~
A
IP65
µA mA
Ω Hz V
COM
A


000
HIKARI HM-1002
V~
OFF
1000
750
200
200
V⎓
A⎓
20
2000 µ
2000 m
20m
200 m
200 m
2000 k
10A
200k
hFE
Ω
20k
2000
200
PNP
NPN
CAT I 1000V
10A
COM
VΩmA


ET-1507B
DC
A
0.000
AUTO APO
SELECT
HOLD
RANGE
MAXMIN
Hz
mA
µA
OFF
NCV
AUTO POWER OFF
TRUE RMS
10A
COM
VΩHzmA


HV
000
NPN
ET-1000
hFE
UNFUSED
10AMAX
OFF
600
200
PNP
2000µ
20m
200m
2000K
200K
20K
2000
200
COM
CAT I 600V


HV
000
DCV
ACV
DCA
OFF
NPN
PNP
BRASFORT
COM

Saídas


ET-2042F
V
DC
00.00
DIGITAL MULTIMETER
SEL
OFF
HOLD
True RMS
mA µA
20A
COM


V
DC
00.00


000
V⎓
OFF
V~
200
20
2000
200
2000k
200k
A⎓
2000µ
20m
200m
10A
hFE
2000
200
Ω
10A⎓
10A MAX - AT 10S
A CADA 15 MIN.
VΩ mA
COM
BRASFORT
Ref.: 8522
CAT II
NPN
PNP
000


FLUKE 115 TRUE RMS MULTIMETER
600.0
HOLD
RANGE
OFF
COM
600.0


Trimips
ET-2507B
TRMS
V
Loz
AC
599.9
BL
0 10 20 30 40 50 60
True RMS
6000 Counts
RANGE
MAX MIN
REL
Hz%
HOLD
SELECT
mV
Hz%
V
µA
OFF
mA
LoZ V~
A
IP65
µA mA
Hz V
COM
A
800mA max
FUSED 1000V max
CAT III 1000V
CAT IV 600V
20A max
FUSED 1000V max
MAX 30sec EACH 15min


TRMS
V
Loz
AC
599.9
BL
0 10 20 30 40 50 60


000
HIKARI HM-1002
V~
OFF
1000
750
200
V⎓
200
A⎓
20
2000 µ
2000 m
20m
200 m
200 m
2000 k
10A
200k
hFE
PNP
Ω
20k
2000
200
CAT I 1000V
NPN
UNFUSED MAX 10 sec EACH 15min
FUSED MAX 200mA 750V~ 1000V⎓
10A
COM
VΩmA


000


ET-1507B
DC
A
0.000
AUTO APO
SELECT
HOLD
RANGE
REL
MAX/MIN
OFF
NCV
AUTO POWER OFF
TRUE RMS
10A
COM
VΩHzmA
CAT III 600V
10sec MAX
10A
MAX
600mA 1000V DC
FUSED 750V AC


DC
A
0.000
AUTO APO


HV
000
NPN
ET-1000
hFE
UNFUSED
10AMAX
OFF
600 V~
200
PNP
2000p
20m
200m
2000K
200K
20K
2000
200
COM
CAT I 600V


HV
000

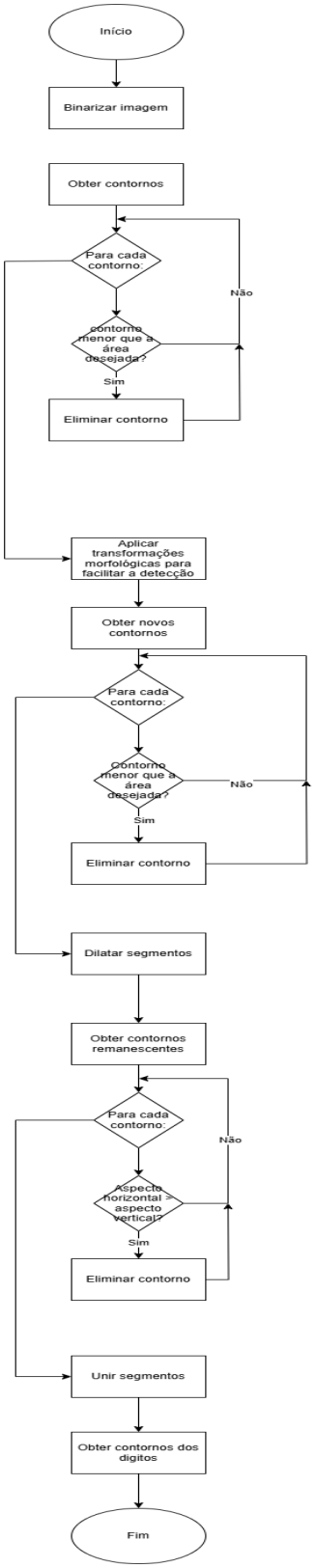
29-06-2025

Visão geral

Foi criada uma função para identificar e isolar as regiões de interesse dos dígitos no display.

Também foram realizados ajustes na obtenção da imagem do display e na limpeza de ruídos.

Fluxograma


Início
Binarizar imagem
Obter contornos
Para cada contorno:
Não
contorno menor que a área desejada?
Sim
Eliminar contorno
Aplicar transformações morfológicas para facilitar a detecção
Obter novos contornos
Para cada contorno:
Não
contorno menor que a área desejada?
Sim
Eliminar contorno
Dilatar segmentos
Obter contornos remanescentes
Para cada contorno:
Não
Aspecto horizontal > aspecto vertical?
Sim
Eliminar contorno
Unir segmentos
Obter contornos dos digitos
Fim

Código

```python
def getDigitRois(img, target_area):
  # Obter a imagem binarizada
  binary_img = getBinaryImage(img, (3, 3))

  # Eliminar contornos muito pequenos ou muito grandes antes de aplicar morfologia
  contours, _ = cv.findContours(binary_img, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)

  kernel = cv.getStructuringElement(cv.MORPH_RECT, (4, 4))
  binary_img = cv.erode(binary_img, kernel, iterations=1)

  for cnt in contours:
    area = cv.contourArea(cnt)

    if area < target_area * 0.09:
      x, y, w, h = cv.boundingRect(cnt)
      cv.rectangle(binary_img, (x, y), (x + w, y + h), (0, 0, 0), -1)

  # Aplicar transformações morfológicas para melhorar a detecção
  kernel = cv.getStructuringElement(cv.MORPH_RECT, (1, 3))
  binary_img = cv.dilate(binary_img, kernel, iterations=1)
  kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
  binary_img = cv.dilate(binary_img, kernel, iterations=1)
  binary_img = cv.morphologyEx(binary_img, cv.MORPH_CLOSE, kernel)

  # Encontrar contornos novamente após as transformações morfológicas
  contours, _ = cv.findContours(binary_img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

  # Eliminar contornos pequenos remanescentes
  for cnt in contours:
    area = cv.contourArea(cnt)
    if area < target_area * 0.325:
      x, y, w, h = cv.boundingRect(cnt)
      cv.rectangle(binary_img, (x, y), (x + w, y + h), (0, 0, 0), -1)

  # Dilatar segmentos
  kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
  binary_img = cv.dilate(binary_img, kernel, iterations=1)

  contours, _ = cv.findContours(binary_img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

  # Eliminar contornos com aspecto de linhas horizontais
  for cnt in contours:
    x, y, w, h = cv.boundingRect(cnt)
    aspect_ratio = float(w) / h
    if aspect_ratio > h / float(w) * 3:
      cv.rectangle(binary_img, (x, y), (x + w, y + h), (0, 0, 0), -1)

  # Unir os segmentos
  kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 7))
  binary_img = cv.dilate(binary_img, kernel, iterations=1)
  cv.imshow('Morphed Image', binary_img)

  # Encontrar contornos dos digitos
  contours, _ = cv.findContours(binary_img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

  digit_rois = []
  for cnt in contours:
    area = cv.contourArea(cnt)
    x, y, w, h = cv.boundingRect(cnt)
    aspect_ratio = float(w) / h
    if target_area * 0.7 <= area <= target_area * 5 and aspect_ratio < h / float(w):
      digit_rois.append(img[y:y+h, x:x+w])
      cv.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
  return digit_rois if digit_rois else None
```

Entradas

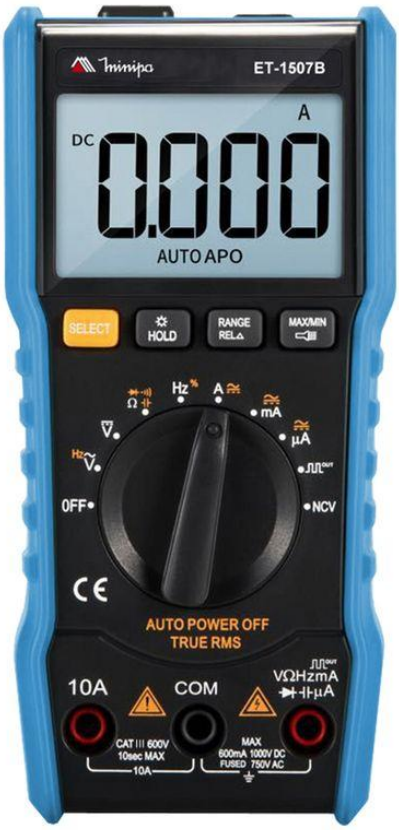

BRASFORT
Ref.: 8522
CAT II
COM
hFE


Minipa
ET-2042F
DC
DIGITAL MULTIMETER
True RMS
COM


FLUKE 115 TRUE RMS MULTIMETER
600.0
V AC
HOLD
MIN MAX
RANGE
OFF
Hz
V
mV
Ω
A
Hz
A
COM
V Ω
10 A FUSED
CAT III 600 V


Minipa
ET-2507B
TRMS
V
LoZ
AC
599.9
BL
0 10 20 30 40 50 60
True RMS
6000 Counts
RANGE
MAX MIN
REL
Hz%
HOLD
OFF
SELECT
Ω
mV
Hz%
V
μA
OFF
mA
LoZ V
A
IP65
μA mA
Ω Hz V
600mA max FUSED 1000V max
CAT III 1000V CAT IV 600V
A
COM
20A max FUSED 1000V max MAX 30sec EACH 15min


000
HIKARI HM-1002
V~
OFF
1000
750
200
200
A⎓
V⎓
20
2000
µ
2000
m
20m
200
m
200
m
10A
2000
k
200k
hFE
PNP
Ω
20k
2000
200
CAT I
1000V
NPN
UNFUSED
MAX 10 sec
EACH 15min
FUSED
MAX 200mA
750V~
1000V⎓
10A
COM
VΩmA


Tmimpo
ET-1507B
DC
A
0.000
AUTO APO
SELECT
HOLD
RANGE
REL
MAX/MIN
Hz%
A
mA
µA
OFF
NCV
AUTO POWER OFF
TRUE RMS
10A
COM
VΩHzmA
CAT III 600V
10sec MAX
10A
MAX
600mA 1000V DC
FUSED 750V AC


HV
000
NPN
Minipa
ET-1000
hFE
UNFUSED
10AMAX
OFF
600
V~
200
PNP
V⎓
2000µ
20m
A⎓
200m
10
2000K
200K
Ω
20K
2000
200
hFE
COM
— VΩmA
600V⎓MAX
600V~MAX
200mA⎓MAX
FUSED
CAT I 600V

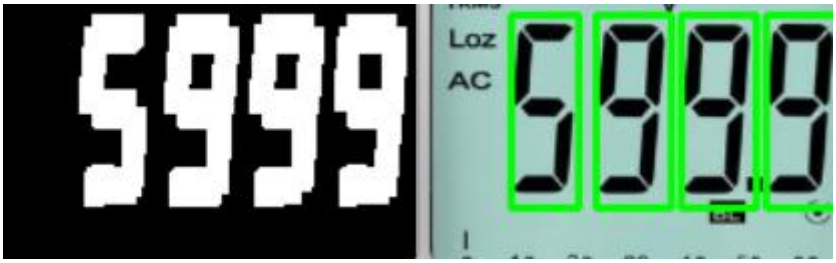

HV
000
DCV
1000
200
20
2000m
200m
2000k
200k
OFF
750
ACV
200
DCA
200µ
2000µ
20m
200m
10A
hFE
20k
2000
Ω
200
NPN
PNP
BRASFORT
10ADC
VΩmA
COM
500Vmax

Saídas


0000
V
DC
0.000


0000
0000


6000
6000


5999
Loz
AC
5999


000
000


0000
A
DC
0000
AUTO APO


HV


HV