TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KỲ MÔN
NHẬP MÔN XỬ LÝ ẢNH SỐ

TRUY XUẤT THỜI GIAN TỪ ẢNH CHỤP ĐỒNG HỒ

Người hướng dẫn: **TS TRỊNH HÙNG CƯỜNG**
Người thực hiện: **PHẠM HUỲNH ANH THƯ - 52000409**
TRẦN NGUYỆT MINH – 52000574
Lớp : **20050401**
Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KỲ MÔN
NHẬP MÔN XỬ LÝ ẢNH SỐ

TRUY XUẤT THỜI GIAN TỪ ẢNH CHỤP ĐỒNG HỒ

Người hướng dẫn: **TS TRỊNH HÙNG CƯỜNG**
Người thực hiện: **PHẠM HUỲNH ANH THƯ - 52000409**
TRẦN NGUYỆT MINH – 52000574
Lớp : **20050401**
Khoá : **24**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023

LỜI CẢM ƠN

Lời đầu tiên chúng em xin gửi lời cảm ơn đến thầy Trịnh Hùng Cường đã tận tình truyền đạt những kiến thức của môn học này, đây không chỉ là nền tảng cho quá trình nghiên cứu làm báo cáo mà còn là hành trang quý báu để chúng em trang bị cho bản thân vốn kiến thức vững chắc.

Bài báo cáo thực hiện trong khoảng thời gian gần 4 tuần. Bước đầu đi vào thực hiện của chúng em còn hạn chế và còn nhiều bỡ ngỡ nên không tránh khỏi những thiếu sót, chúng em rất mong nhận được những ý kiến đóng góp quý báu của Thầy để chúng em được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kiến thức của mình trong môn học này.

Chúng em xin chân thành cảm ơn!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS Trịnh Hùng Cường;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 24 tháng 10 năm 2023

Tác giả

(ký tên và ghi rõ họ tên)

Phạm Huỳnh Anh Thư

Trần Nguyệt Minh

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(ký và ghi họ tên)

TÓM TẮT

Bài báo cáo xoay quanh vấn đề truy xuất thời gian tử ảnh chụp đồng hồ kim trao tường.

Báo cáo này có sử dụng các kỹ thuật như tiền xử lý ảnh, xác định góc của kim giờ, kim phút, và kim giây. Sau đó, thông qua các công thức toán học, chuyển đổi thông tin góc kim thành các giá trị thời gian cụ thể, bao gồm giờ, phút, và giây.

MỤC LỤC

DANH MỤC CÁC HÌNH ẢNH

DANH MỤC HÌNH

CHƯƠNG 1: PHƯƠNG PHÁP GIẢI QUYẾT ĐỀ TÀI

1.1 Các nội dung cần thiết

- OpenCV: thư viện chính được sử dụng xuyên suốt bài báo cáo, một thư viện cung cấp đa dạng chức năng để xử lý ảnh: đọc, viết, tìm contour, tìm đường thẳng,…
- Xác định góc trong đường tròn dựa vào toán học
- Xác định khoảng cách 1 điểm đến tâm đường tròn
- Dựa vào góc đã xác định áp dụng vào đường tròn 360^o để tìm chính xác thời gian của từng kim đồng hồ

1.2 Các hàm thực hiện

1.2.1 Đọc ảnh đầu vào

```
7  # ---------- 2. READ INPUT IMAGES FROM FOLDER ----------
8  def read_from_path(path):
9      images = []
10     for filename in os.listdir(path):
11         if filename.endswith(".jpg") or filename.endswith(".png") or filename.endswith(".jpeg"):
12             img_path = os.path.join(path, filename)
13             img = cv.imread(img_path)
14             if img is not None:
15                 images.append(img)
16     return images
17
18 # Specify the path to read input images
19 path = os.getcwd() + "/input"
20 images = read_from_path(path)
```

Hình 1 Hàm read_from_path(path)

Hàm ***read_from_path(path)***:

- Nhận vào đường dẫn ***path*** đến thư mục tên input chứa các ảnh dưới dạng .png, .jpeg, .jpg
- Dùng hàm ***cv.imread()*** để đọc ảnh dưới dạng ảnh màu sau đó lưu vào list ***images***
- List images sẽ chứa các ảnh gốc để dùng đến các bước tiếp theo

1.2.2 Tiền xử lý ảnh

```python
# ---------- 5. PRE PROCESSING ----------
def pre_processing(img_list):
    img_gray = []
    img_pre = []
    for img in img_list:
        gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        ####
        img_gray.append(gray)
        ####
        thresh = cv.threshold(gray, img.mean(), 255, cv.THRESH_BINARY)[1]
        thresh = 255 - thresh

        img_pre.append(thresh)

    return img_gray, img_pre
images_gray, images_pre = pre_processing(images)
```

Hình 2 Hàm pre_processing(img_list)

Hàm ***pre_processing(img_list)***:

- Nhận vào list ***images*** từ bước đọc ở trên và trả về list chứa ảnh xám ***images_gray*** và chứa ảnh nhị phân ***images_pre***
- Hàm sẽ dùng vòng lặp duyệt từng ảnh trong list images
- Dùng hàm ***cv.cvtColor()*** để chuyển ảnh gốc sang ảnh xám và lưu vào list ảnh xám
- Dùng hàm ***cv.threshold*** với enumerator là ***THRESH_BINARY*** với công thức:

$$\text{dst}(x, y) = \begin{cases} 255 & \text{if } \text{src}(x, y) > \text{thresh} = \text{img.mean}() \\ 0 & \text{otherwise} \end{cases}$$

  Nếu pixel của điểm ảnh đang xét > mean của ảnh thì đổi thành pixel trắng

  Ngược lại đổi thành pixel đen

- Sau đó lưu ảnh vào ***list images_pre*** và trả kết quả

1.2.3 Tìm thời gian

```python
# Sort contours by area
def sort_by_area(li):
    return li[1]

```

Hình 3 Hàm sort_by_area()

Hàm ***sort_by_area()*** sẽ là key của hàm sort bằng việc sort phần tử ở vị trí [1] trong list contours chính là diện tích của từng contour

```python
# Get farest point
def get_true_order(order, center):
    # Check which x,y is the farest
    x_n = order[1][0][0]
    y_n = order[1][0][1]
    x_f = order[1][1][0]
    y_f = order[1][1][1]

    # Cal length for each (a,b) --> (c, d) to find the farest point
    length1 = np.sqrt((x_n - center[0]) ** 2 + (y_n - center[1]) ** 2)
    length2 = np.sqrt((x_f - center[0]) ** 2 + (y_f - center[1]) ** 2)

    if length1 > length2:
        return ((x_f, y_f), (x_n, y_n))
    else:
        return ((x_n, y_n), (x_f, y_f))
```

Hình 4 Hàm get_true_order(order, center)

Hàm ***get_true_order(order, center)*** để tìm ra vị trí (x1, y1) là điểm gần tâm và (x2, y2) là điểm xa tâm của từng kim so với tâm đồng hồ:

- Công thức khoảng cách từ 1 điểm đến tâm
- $\sqrt{(x - xo)^2 + (y - yo)^2}$
- Dựa vào công thức ta xác định lần lượt length1 (từ (x1, y1) -> tâm) và length2 (từ (x2, y2) -> tâm)
- Nếu length1 > length2 -> (x1, y1) là điểm xa tâm do đó thứ tự đúng là (x2, y2) -> (x1, y1)
- Ngược lại thứ tự đúng là (x1, y1) -> (x2, y2)

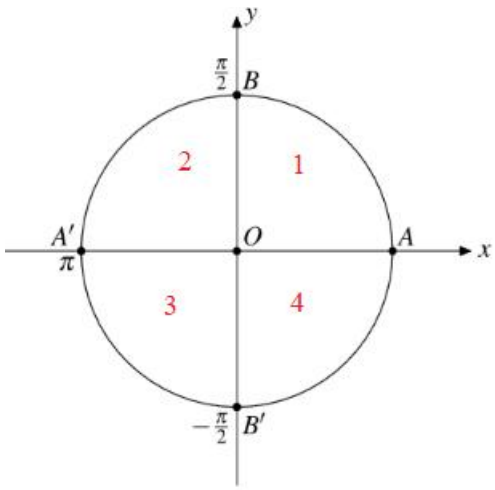
```python
# Find angle of each hand --> for get exact time
def find_angle(hand,center):
    x_h = hand[0]
    y_h = hand[1]
    x_c = center[0]
    y_c = center[1]

    x_diff = x_h - x_c
    y_diff = y_h - y_c
    x_diff = float(x_diff)
    y_diff = float(y_diff)

    if(x_diff*y_diff > 0):
        if(x_diff >= 0 and y_diff > 0):
            return np.pi-np.arctan(x_diff/y_diff)
        elif(x_diff <= 0 and y_diff < 0):
            return 2*np.pi - np.arctan(x_diff/y_diff)
    elif(x_diff*y_diff < 0):
        if(y_diff >= 0 and x_diff < 0):
            return (3*np.pi)/4 + np.arctan(x_diff/y_diff)
        elif(y_diff <= 0 and x_diff > 0):
            return -np.arctan(x_diff/y_diff)
```

Hình 5 Hàm find_angle(hand, center)

Hàm ***find_angle(hand, center)*** nhận vào tọa độ (x2, y2) là điểm xa tâm nhất cửa từng kim đồng hồ và tọa độ (x, y) của tâm contour và trả về góc của từng kim so với tâm dưới dạng radian:



Hình 6 Ảnh đường tròn

- Tính hiệu giữa (x2, y2) so với tâm (x, y): x_{diff} ***và*** y_{diff}
- Nếu tích của x_{diff} và y_{diff} là ***dương*** thì điểm kim đồng hồ sẽ nằm ở góc phần tư thứ 1 hoặc 3

- Nếu tích của x_{diff} và y_{diff} là ***âm*** thì điểm kim đồng hồ sẽ nằm ở góc phần tư thứ 2 hoặc 4

```
90  # Find hour by angle
91  def get_hour(angle):
92      hour = angle//30
93      if(hour == 0):
94          return 12
95      else:
96          return int(hour)
97
98  # Find min by angle
99  def get_min(angle):
100     min = angle/(np.rad2deg(2*np.pi))*60
101     return int(min)
102
103 # Find sec by angle
104 def get_sec(angle):
105     sec = angle/(np.rad2deg(2*np.pi))*60
106     return int(sec)
107
```

Hình 7 Hàm get_hour(angle), get_min(angle), get_sec(angle)

Các hàm ***get_hour(angle)***, ***get_min(angle)*** và ***get_sec(angle)*** nhận vào góc đã xác định ở trên của từng kim

```
130 # Get time
131 def findTime(img, imgO):
132     try:
133         # Find the clock by the biggest contour and wrap around it with green
134         contours, hierarchy = cv.findContours(img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
135         cv.drawContours(imgO, clock_contour, -1, (0, 255, 0), 2)
136
```

Hình 8 Tìm contour đồng hồ

- Xác định bằng RETR_EXTERNAL để chỉ lấy contour bên ngoài

```
137         # Find the center of clock contour and circle it with red dot
138         clock_contour = max(contours, key = cv.contourArea)
139         (x, y), radius = cv.minEnclosingCircle(clock_contour)
140         center = (int(x), int(y))
```

Hình 9 Tìm tâm, bán kính đồng hồ

- Dùng hàm max() với key = contourArea để lấy ra contour có diện tích lớn nhất
- Dùng hàm minEnclosingCircle() để lấy ra tọa độ tâm center và bán kính đồng hồ radius

```
142         # Find all the contours then sort it by dec
143         contoursL,_ = cv.findContours(img, cv.RETR_LIST, cv.CHAIN_APPROX_SIMPLE)
144         contour_list = []
145         i = 0
146         for cnt in contoursL:
147             area = cv.contourArea(cnt)
148             contour_list.append((i, area))
149             i = i + 1
150 
151         # Sort contours by area
152         contour_list.sort(key = sort_by_area, reverse=True)
153 
154         # Get the three contour inside the clock contour --> mask the hand area --> to detect line later
155         handMasked = np.zeros_like(img)
156         threeMasked = contour_list[2][0]
157         cv.drawContours(handMasked,[contoursL[threeMasked]],0,(1),-1)
```

Hình 10 Tìm contour các kim

- Dòng 143: lúc này tìm contour bằng RETR_LIST để trả về list các contour thay vì chỉ contour bên ngoài
- Dòng 144 -> 149: khởi tạo một list contour_list với 2 phần tử là tọa độ và diện tích các contour vừa tìm
- Dòng 152: sort list trên theo diện tích contour theo chiều giảm dần
- Dòng 155 -> 157: tạo một mask của 3 contour đầu tiên sau khi sort, mask với màu đen

```
159         # Convert that hand back to white for clear view
160         handMasked = (255 * handMasked).clip(0,255).astype(np.uint8)
161         kernel = np.ones((1,2),np.uint8)
162         handMasked = cv.erode(handMasked, kernel, 1)
163         handMasked = cv.morphologyEx(handMasked, cv.MORPH_CLOSE, np.ones((1,1),np.uint8))
164         erode_closing = handMasked.copy()
165         # handMasked = cv.dilate(handMasked, kernel, 1)
166         handMasked = skeletonize(handMasked)
167         handMasked = (255 * handMasked).clip(0,255).astype(np.uint8)
```

Hình 11 Xử lý ảnh các kim

- Dòng 160 -> 164: tiến hành erode và closing ảnh để lấy được ảnh của kim đồng hồ ít nhiễu và gọn gàng hơn
- Dòng 166: dùng hàm skeletonize() để lấy cấu trúc xương sống từng kim đồng hồ

```
169         # Find the line hand on the masked img
170         lines_list = {}
171         lines = cv.HoughLinesP(
172                     handMasked,
173                     1,
174                     np.pi/180,
175                     threshold=35,
176                     minLineLength=50,
177                     maxLineGap=radius
178                     )
179
180         count_line = 0
181         if lines is not None:
182             count_line = len(lines_list)
183             print("LINES: ", len(lines))
184             for points in lines:
185                 x1, y1, x2, y2 = points[0]
186                 length = np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
187                 if length > 60:
188                     # Draw hand line with red
189                     # cv.line(imgO,(x1,y1),(x2,y2),(0,0,255),2)
190
191                     lines_list[length] = ((x1, y1), (x2, y2))
192
193             # Find the hour, min, sec by it length: hour < min < sec
194             lines_list = sorted(lines_list.items())
195             top3 = lines_list[:3]
```

Hình 12 Tìm tọa độ các kim

- Dùng ***HoughLineP()*** để xác định đường thẳng ở ảnh đã xử lý ở trên với min vote = 35
- Dòng 170 -> 191: khởi tạo một dict lines_list để chứa thông tin của đường thẳng, key = length của đường thẳng, value = cặp tọa độ
- Dòng 194: sort list trên theo chiều giảm dần theo key
- Dòng 195: lấy 3 ra cặp tọa độ đầu tiên của các kim đồng hồ

```
197         # The farest point of each hand --> length, ((x1, y1), (x2, y2)) --> (x2, y2)
198         hour_hand = (get_true_order(top3[0], center))
199         min_hand = (get_true_order(top3[1], center))
200         sec_hand = (get_true_order(top3[2], center))
```

Hình 13 Tọa độ từng kim

- Lấy ra lần lượt tọa độ của kim giờ, phút, giây (vì chiều dài kim giờ < phút < giây)
- Xác định chính xác điểm gần tâm và xa tâm của từng kim đồng hồ bằng hàm get_true_order()

```
231             # Draw each hand wrapped by rectangle
232             cv.rectangle(imgO, hour_hand[0], hour_hand[1],(0, 255, 0),2) # hour --> green
233             cv.rectangle(imgO, min_hand[0], min_hand[1],(255, 0, 0),2) # min --> blue
234             cv.rectangle(imgO, sec_hand[0], sec_hand[1],(0, 255, 255),2) # sec --> yellow
235
```

Hình 14 Vẽ hình chữ nhật quanh kim

- Sau khi có tọa độ của từng kim đồng hồ tiến hành vẽ hình chữ nhật bao quanh từng kim với các màu: kim giờ -> xanh lá, kim phút -> xanh dương, kim giây -> vàng

```
253             h, m, s = "", "", ""
254             if hour < 10:
255                 h = '0' + str(hour)
256             else:
257                 h = str(hour)
258             if min < 10:
259                 m = '0' + str(min)
260             else:
261                 m = str(min)
262             if sec < 10:
263                 s = '0' + str(sec)
264             else:
265                 s = str(sec)
266
267             time = h + ":" + m + ":" + s
268             cv.putText(imgO,time, (50,50), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2, cv.LINE_AA)
269
270             cv.circle(imgO, center, 2, (0,0,255), 2)
271             cv.circle(imgO, true_center, 2, (100, 10, 79), 5)
```

Hình 15 Vẽ thời gian lên ảnh

- Viết lên ảnh với định dạng giờ:phút:giây vừa tìm bằng màu đỏ và ở tọa độ (50, 50)

1.2.4 Thể hiện kết quả

```
22  # ---------- 3. SHOW IMAGES ----------
23  def show(img_list):
24      i = 1
25      for img in img_list:
26          cv.imshow('Image ' + str(i), img)
27          i = i + 1
28      cv.waitKey(0)
29      cv.destroyAllWindows()
30
```

Hình 16 Hàm show(img_list)

1.2.5 Lưu kết quả

```python
# ---------- 4. SAVE IMAGES ----------
def write_to_path(images_list):
    count = 1
    path = os.getcwd() + "/output/"
    for img in images_list:
        cv.imwrite(path + 'output' + str(count) + '.png', img)
        count += 1

def write_to_path_precessing(path, images_list, step):
    count = 1
    for img in images_list:
        cv.imwrite(path + step + str(count) + '.png', img)
        count += 1

```

Hình 17 Hàm write_to_path(images_list)

```python
# Save the output
pathP = os.getcwd() + "/output/"
write_to_path_precessing(pathP + 'gray/', images_gray, 'gray')
write_to_path_precessing(pathP + 'thresh/', images_pre, 'thresh')
write_to_path_precessing(pathP + 'erode_closing/', images_erode_closing, 'erode_closing')
write_to_path_precessing(pathP + 'skeletonize/', images_skeletonize, 'skeletonize')
write_to_path(images_clock)
show(images_clock)
```

Hình 18 Lưu các ảnh vào thư mục output

CHƯƠNG 2: CHẠY THỰC NGHIỆM VÀ KẾT QUẢ

1.1 Load các thư viện cần thiết

```python
# ---------- 1. LOAD LIBRABRY ----------
import os
import cv2 as cv
import numpy as np
from skimage.morphology import skeletonize

```

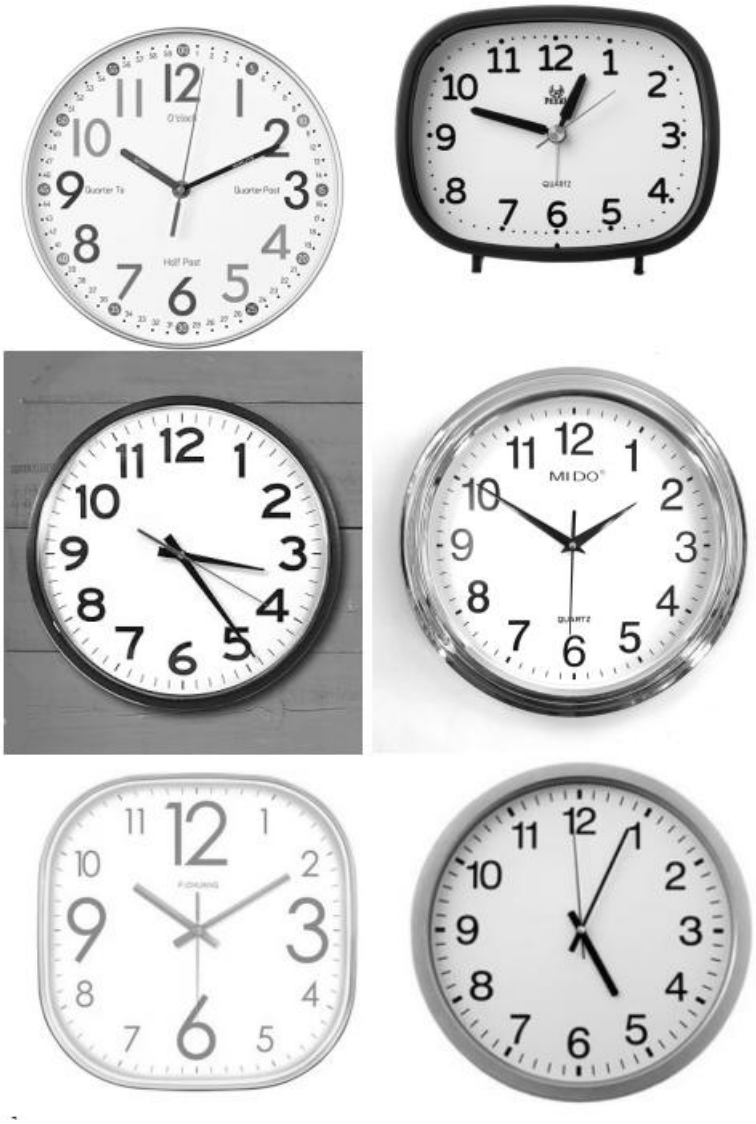
1.2 Đọc ảnh đầu vào và show ảnh

Ở đây sẽ lấy ví dụ với 6 ảnh có thời gian khác nhau:

Hình 19 Các ảnh gốc

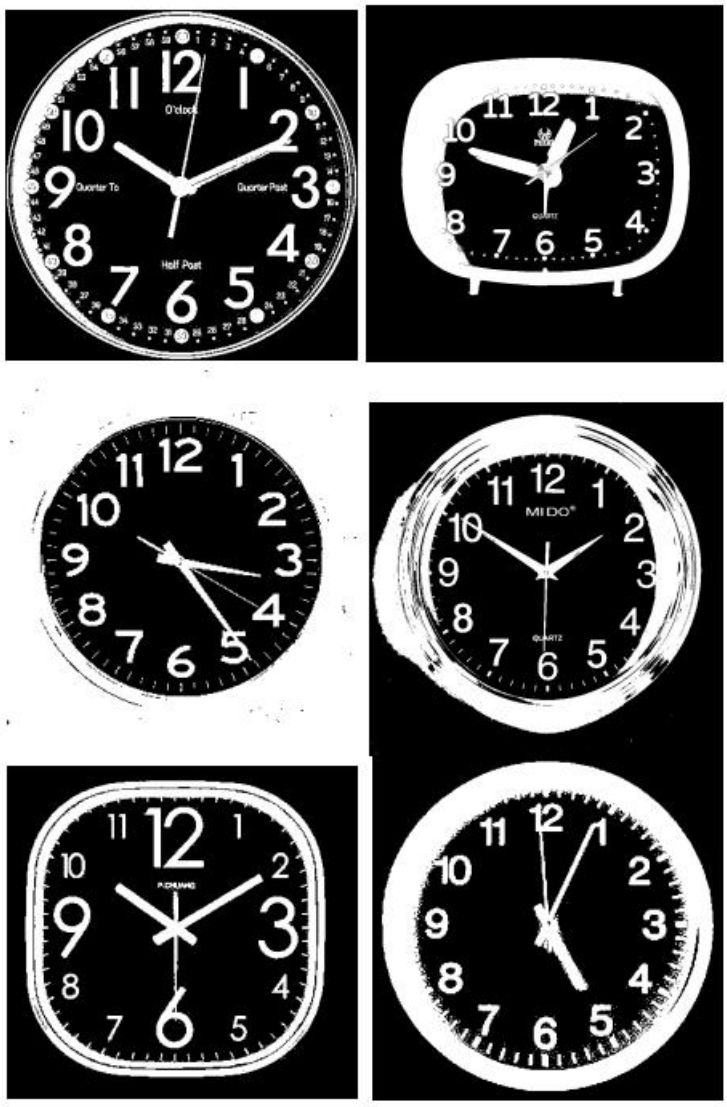
Tiền xử lý ảnh

- Hàm pre_processing(img_list): nhận vào một list chứa các ảnh đầu vào images ở trên để tiến hành xử lý từng ảnh
- Chuyển đổi ảnh thành ảnh xám



Hình 20 Các ảnh xám

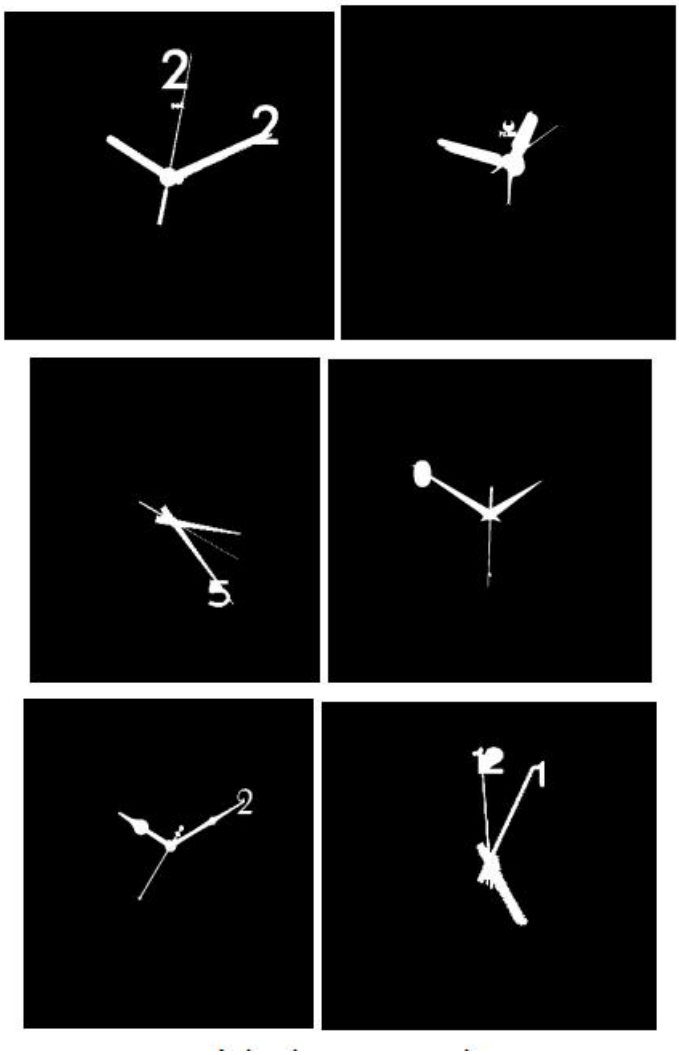
- Chuyển ảnh xám thành ảnh nhị phân



Hình 21 Các ảnh nhị phân

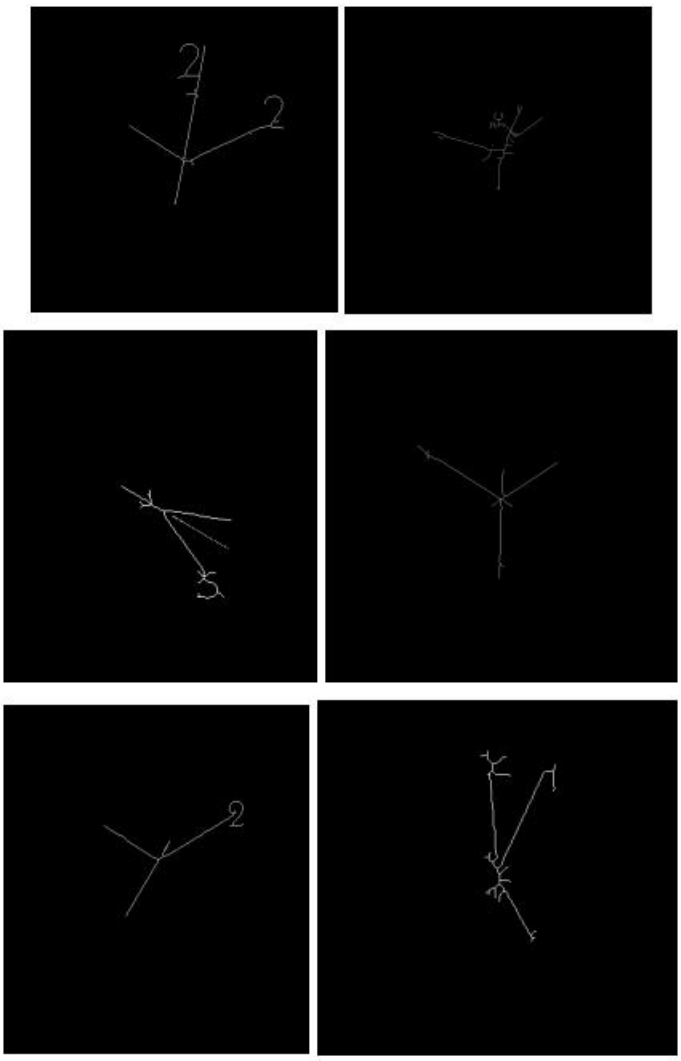
1.3 Tìm tọa độ các kim đồng hồ

- Ảnh erode và closing để lấy được ảnh của kim đồng hồ ít nhiễu và gọn gàng hơn



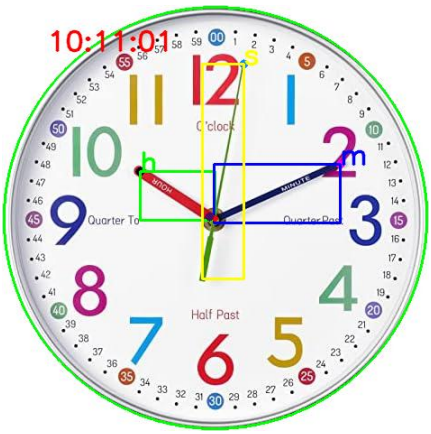
Hình 22 Các ảnh kim sau erode và closing

- Ảnh đã được skeletonize() để lấy cấu trúc xương sống của từng kim



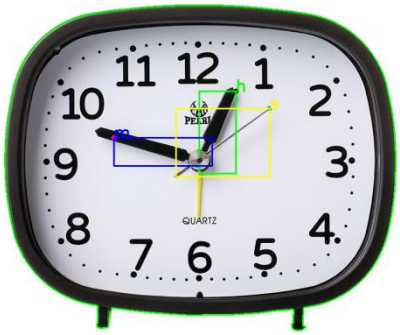
Hình 23 Các ảnh kim sau skeletonize

1.4 Kết quả cuối cùng

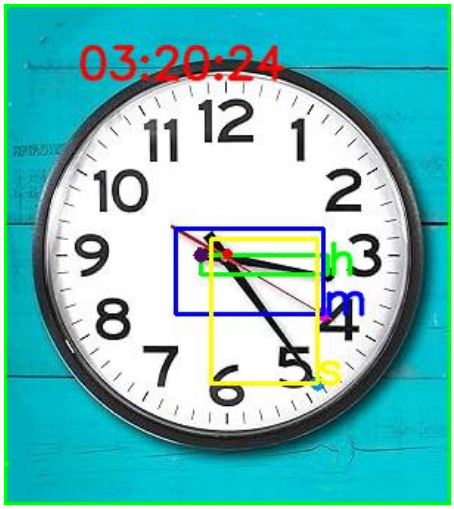


Hình 24 Kết quả 1

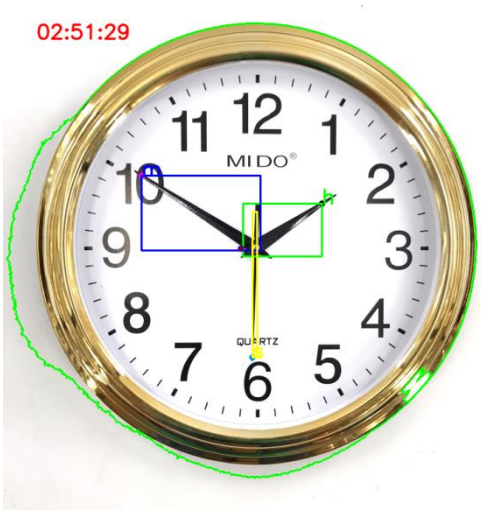
12:47:08



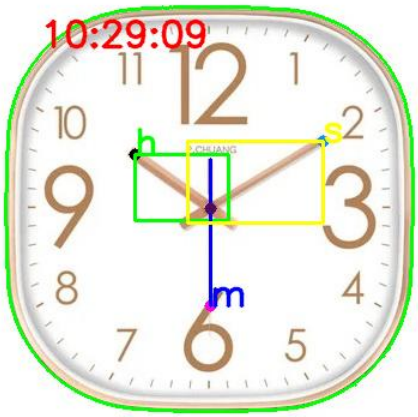
Hình 25 Kết quả 2



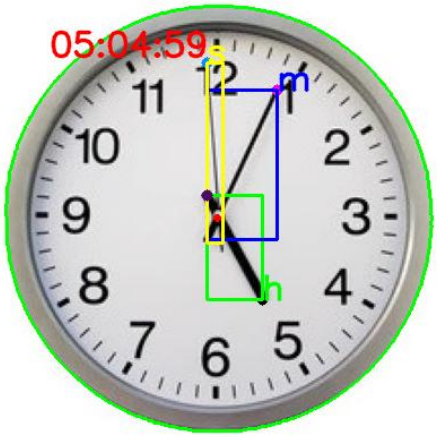
Hình 26 Kết quả 3



Hình 27 Kết quả 4



Hình 28 Kết quả 5



Hình 29 Kết quả 6

TÀI LIỆU THAM KHẢO

Tiếng Anh

Bài báo cáo sử dụng các kiến thức từ những bài thực hành trên lớp.