

Final Project Card Detection with "Minuman" games

M. Radhito Bil Atho 5024211056

Pengolahan Citra Gambar dan Video (A)
Teknik Komputer
Institut Teknologi Sepuluh Nopember

Desember 19, 2023

1 Pendahuluan

Dalam Pengolahan Citra gambar dan video, dapat memproses sebuah gambar sedemikian rupa agar mudah dikenali. Dalam konteks ini, Pendekripsi kartu yang diambil melalui camera akan melewati serangkaian proses pengolahan citra. Pengolahan Citra pada kartu dilakukan mengatur warna pada kartu, mendekripsi rupa atau wujud tepi kartu, filterisasi, dan lainnya. Dari hasil pengolahan citra tersebut, dapat kita kombinasikan dengan metode CNN dari machine learning agar komputer mengetahui kartu yang sedang ingin didekripsi. Model CNN akan mengambil serangkaian dataset terlebih dahulu sebagai peninjauan atau prediksi terhadap kartu yang sedang didekripsi.

Dengan hasil deteksi tersebut, dapat digunakan serangkaian algoritma untuk berjalannya game minuman ini. Hasil dari deteksi akan dijadikan objek secara real-time dan sebagai tolak ukur win condition dengan mengatur aturan-aturan pada permainan Minuman ini. ov

2 Dasar Teori

OpenCV atau Open Source Computer Vision Library digunakan sebagai open source untuk dapat memudahkan membaca atau memahami dan menganalisa gambar maupun video dengan fungsi algoritma siap pakai. Dalam konteks ini, OpenCV juga digunakan untuk menganalisa dan memanipulasi citra video dengan melakukan filtering dan pengaplikasian berbagai operasi filter pada citra. pengolahan citra ini dilakukan agar citra dapat mudah dikenali dengan mengatur warna, tipe, dan ukuran citra. OpenCV juga dijadikan media visual yang akan menampilkan semuanya. Hasil dari filter citra akan dilakukan training.

Training dilakukan dengan kumpulan data atau citra yang digunakan untuk melatih model machine learning. Dataset kartu harus mencakup gambar kartu dengan berbagai variasi pose dari bentuk kartu untuk peningkatan akurasi. Digunakan Convolutional Neural Network (CNN) sebagai jaringan saraf tiruan untuk menjalankan proses terkait pengolahan citra. Hal ini dilakukan dengan CNN mempelajari pola visual dari kartu.

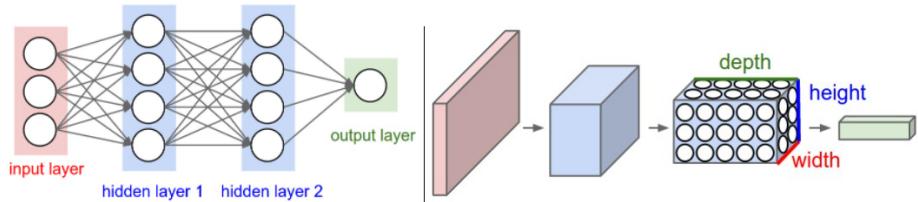


Figure 1: Cara Kerja CNN

Dalam CNN memiliki layer - layer yang bertanggung jawab untuk mengekstrak visual dari data citra. Convolutional Layer untuk mengekstrak data dengan menerapkan filter konvolusi. Pooling Layer mengurangi ukuran representasi fitur dari data citra. Fully Connected Layer digunakan untuk mengklasifikasikan data citra.

3 Source Code

Pada Project ini, digunakan 6 file Source code yang terdiri dari builddataset, KlasifikasiCNN, Traindata, main, menu, dan close.

3.1 Build Dataset

```

1 import cv2
2 import os
3 import datetime
4 import time
5 import numpy as np
6
7 # Untuk penamaan semua class di Model ML
8 cardName = [
9     "Kartu Tutup",
10    "Keriting Dua",
11    "Keriting Tiga",
12    "Keriting Empat",
13    "Keriting Lima",
14    "Keriting Enam",
15    "Keriting Tujuh",
16    "Keriting Delapan",
17    "Keriting Sembilan",
18    "Keriting Sepuluh",

```

```

19     "Keriting Jack",
20     "Keriting Queen",
21     "Keriting King",
22     "Keriting Ace",
23     "Hati Dua",
24     "Hati Tiga",
25     "Hati Empat",
26     "Hati Lima",
27     "Hati Enam",
28     "Hati Tujuh",
29     "Hati Delapan",
30     "Hati Sembilan",
31     "Hati Sepuluh",
32     "Hati Jack",
33     "Hati Queen",
34     "Hati King",
35     "Hati Ace",
36     "Wajik Dua",
37     "Wajik Tiga",
38     "Wajik Empat",
39     "Wajik Lima",
40     "Wajik Enam",
41     "Wajik Tujuh",
42     "Wajik Delapan",
43     "Wajik Sembilan",
44     "Wajik Sepuluh",
45     "Wajik Jack",
46     "Wajik Queen",
47     "Wajik King",
48     "Wajik Ace",
49     "Sekop Dua",
50     "Sekop Tiga",
51     "Sekop Empat",
52     "Sekop Lima",
53     "Sekop Enam",
54     "Sekop Tujuh",
55     "Sekop Delapan",
56     "Sekop Sembilan",
57     "Sekop Sepuluh",
58     "Sekop Jack",
59     "Sekop Queen",
60     "Sekop King",
61     "Sekop Ace",
62 ]
63
64 # Kita mulai dari index 0
65 cardNameIndex = 0
66
67 # Fungsi penamaan file berdasarkan waktu pengambilan
68 def GetFileName():
69     x = datetime.datetime.now()
70     s = x.strftime('%Y-%m-%d-%H%M%S%f')
71     return s
72
73 # Fungsi menentukan luas area dari 4 titik
74 def polygon_area(points):

```

```

75     points = np.vstack((points, points[0]))
76     area = 0.5 * np.abs(np.dot(points[:, 0], np.roll(
77         points[:, 1], 1)) - np.dot(points[:, 1], np.roll(points[:, 0], 1)))
78     return area
79
80 # Fungsi cek dan pembuatan direktori
81 def CreateDir(path):
82     ls = []
83     head_tail = os.path.split(path)
84     ls.append(path)
85     while len(head_tail[1])>0:
86         head_tail = os.path.split(path)
87         path = head_tail[0]
88         ls.append(path)
89         head_tail = os.path.split(path)
90     for i in range(len(ls)-2,-1,-1):
91         sf = ls[i]
92         isExist = os.path.exists(sf)
93         if not isExist:
94             os.makedirs(sf)
95
96 # Fungsi pembuatan dataset
97 def CreateDataSet(sDirektoriData,sKelas,NoKamera,
98 FrameRate):
99     global cardName, cardNameIndex
100
101    cap = cv2.VideoCapture(NoKamera)
102    TimeStart = time.time()
103
104    # Mengetahui limit detik dalam pengambilan gambar
105    saveTimeLimit = time.time()
106
107    # Status record
108    isSaving = False
109
110    while cap.isOpened():
111        success, frame = cap.read()
112
113        # Membuat direktori dataset sesuai label
114        sDirektoriKelas = sDirektoriData+"/"+cardName[
115            cardNameIndex]
116        CreateDir(sDirektoriKelas)
117
118        if not success:
119            print("Ignoring empty camera frame.")
120            continue
121
122        isDetected = False
123
124        # Image Processing untuk filter citra dengan edge
125        detection
126        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
127        blur = cv2.GaussianBlur(gray, (5, 5), 0)
128        canny = cv2.Canny(blur, 100, 150)
129        cv2.imshow("2. Threshold", canny)

```

```

126
127
128         # Mengambil data dari komponen yang terhubung atau
129         # garis dengan ukuran yang diinginkan
130         totalLabels, label_ids, values, centroid = cv2.
131         connectedComponentsWithStats(canny, 4, cv2.CV_32S)
132         bigIndex = []
133         for i in range(totalLabels):
134             hw = values[i,2:4]
135             if (70<hw[0]<200 and 150<hw[1]<300):
136                 bigIndex.append(i)
137
138         # Dari garis yang telah diketahui, diberi gambar
139         persegi (contour)
140         for i in bigIndex:
141             topLeft = values[i,0:2]
142             bottomRight = values[i,0:2]+values[i,2:4]
143             frame = cv2.rectangle(frame, topLeft,
144             bottomRight, color=(0,0,255), thickness=3)
145             # Disini ada break, yg berarti kita cuma
146             ngambil 1 item doang
147             break
148         cv2.imshow("4. Hasil habis dikotakin", frame)
149
150         # Gambar yang berada dalam contour akan diambil (
151         crop)
152         for i in bigIndex:
153             topLeft = values[i,0:2]
154             bottomRight = values[i,0:2]+values[i,2:4]
155             cardImage = canny[topLeft[1]:bottomRight[1],
156             topLeft[0]:bottomRight[0]]
157             cv2.imshow('5. Hasil dari cardImage',
158             cardImage)
159             break
160
161         # Simpan gambar yang dicrop sesuai berjalannya
162         waktu
163         TimeNow = time.time()
164         if TimeNow-TimeStart>1/FrameRate:
165             sfFile = sDirektoriKelas+"/"+.GetFileName()
166             if isSaving and len(bigIndex) > 0:
167                 cv2.imwrite(sfFile+'.jpg', cardImage)
168             TimeStart = TimeNow
169
170             cv2.putText(frame, "Nama Kartu yg direkam:", (0,
171             30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
172             cv2.putText(frame, f"{cardNameIndex+1}. " +
173             cardName[cardNameIndex], (0, 70), cv2.FONT_HERSHEY_SIMPLEX
174             , 0.8, (0, 0, 255), 2)
175             cv2.putText(frame, "Tekan spasi untuk mulai
176             record", (0, 400), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,
177             0,0), 2)
178
179             if isSaving:

```

```

167             cv2.putText(frame, "Record", (0, 30), cv2.
168 FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
169         else:
170             saveTimeLimit = time.time()
171
172         cv2.imshow("Tampilan akhir", frame)
173
174         key = cv2.waitKey(5)
175
176         # Tekan spasi untuk mulai rekam/menyimpan gambar
177         if key == 32:
178             isSaving = not isSaving
179
180         # Kalo udah lebih dari 8 detik, record/
181         # penyimpanan foto selesai
182         if time.time() - saveTimeLimit >= 8:
183             cardNameIndex += 1
184             isSaving = False
185
186         if key & 0xFF == 27:
187             break
188         cap.release()
189         cv2.destroyAllWindows()
190
191         DirektoriDataSet = "dataset"
192         CreateDataSet(DirektoriDataSet, " ", NoKamera=1,
FrameRate=20)
193
194

```

Source Code 1: Membuat Dataset

3.2 Klasifikasi CNN

```

1 import os
2 from keras.models import load_model
3 import cv2
4 import numpy as np
5 from keras.layers import Input, Dense
6 from keras.layers import Conv2D, MaxPooling2D, Flatten
7 from keras.models import Model
8 import matplotlib.pyplot as plt
9 from datetime import datetime
10 from numpy import expand_dims
11 from keras.utils import load_img
12 from keras.utils import img_to_array
13 from keras.preprocessing.image import ImageDataGenerator
14 from matplotlib import pyplot
15
16 def LoadCitraTraining(sDir, LabelKelas):
17     JumlahKelas=len(LabelKelas)
18     TargetKelas = np.eye(JumlahKelas)
19
20     X=[] #Menampung Data Citra
21     T=[] #Menampung Target

```

```

22     for i in range(len(LabelKelas)):
23         #Membaca file citra di setiap direktori data set
24         DirKelas = os.path.join(sDir, LabelKelas[i])
25         files = os.listdir(DirKelas)
26         for f in files:
27             ff=f.lower()
28             print(f)
29             #memilih citra dengan extensi jpg,jpeg,dan png
30             if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.
31             endswith('.png')):
32                 NmFile = os.path.join(DirKelas,f)
33                 img= np.double(cv2.imread(NmFile ,1))
34                 img=cv2.resize(img,(128,128))
35                 img= np.asarray(img)/255
36                 img=img.astype('float32')
37
38                 X.append(img)
39                 T.append(TargetKelas[i])
40
41     #Mengubah List Menjadi numpy array
42     X=np.array(X)
43     T=np.array(T)
44     X=X.astype('float32')
45     T=T.astype('float32')
46     return X,T
47
48 def ModelDeepLearningCNN(JumlahKelas):
49     input_img = Input(shape=(128, 128, 3))
50     x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
51     x = MaxPooling2D((2, 2), padding='same')(x)
52     x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
53     x = MaxPooling2D((2, 2), padding='same')(x)
54     x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
55     x = Flatten()(x)
56     x = Dense(100,activation='relu')(x)
57     x=Dense(JumlahKelas,activation='softmax')(x)
58     ModelCNN = Model(input_img, x)
59     ModelCNN.compile(loss='categorical_crossentropy',
60                       optimizer='adam', metrics=['accuracy'])
61     return ModelCNN
62
63 def TrainingCNN(JumlahEpoh,DirektoriDataSet,LabelKelas,
64                  NamaFileBobot):
65     #Membaca Data training dan label Kelas
66     X,D=LoadCitraTraining(DirektoriDataSet,LabelKelas)
67     JumlahKelas = len(LabelKelas)
68     #Membuat Model CNN
69     ModelCNN =ModelDeepLearningCNN(JumlahKelas)
70     #Trainng
71     history=ModelCNN.fit(X, D,epochs=JumlahEpoh,shuffle=True)
72     #Menyimpan hasil learning
73     ModelCNN.save(NamaFileBobot)

```

```

72     return ModelCNN,history
73
74
75 def Klasifikasi(DirDataSet,DirKlasifikasi,LabelKelas,ModelCNN
76 =[]):
77 #Menyiapkan Data input
78 X=[]
79 ls = []
80 DirKelas = DirDataSet+"/"+DirKlasifikasi
81 print(DirKelas)
82 files = os.listdir(DirKelas)
83 n=0
84 for f in files:
85     ff=f.lower()
86     print(f)
87     if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.
88 endswith('.png')):
89         ls.append(ff)
90         NmFile = os.path.join(DirKelas,f)
91         img= cv2.imread(NmFile,1)
92         img=cv2.resize(img,(128,128))
93         img= np.asarray(img)/255
94         img=img.astype('float32')
95         X.append(img)
96
97 X=np.array(X)
98 X=X.astype('float32')
99
100
101 LKlasifikasi=[]
102 LKelasCitra =[]
103 n = X.shape[0]
104 for i in range(n):
105     v=hs[i,:]
106     if v.max()>0.5:
107         idx = np.argmax(np.where( v == v.max()))
108         LKelasCitra.append(LabelKelas[idx])
109     else:
110         idx=-1
111         LKelasCitra.append("—")
112 #-----akhir if
113 LKlasifikasi.append(idx)
114 #-----akhir for
115 LKlasifikasi = np.array(LKlasifikasi)
116 return ls, hs, LKelasCitra
117
118 def LoadModel(sf):
119     ModelCNN=load_model(sf)
120     return ModelCNN
121

```

Source Code 2: Modul Klasifikasi CNN

```
1 import KlasifikasiCNN as mCNN
```

```
2
3 # Nama folder datasetnya
4 DirektoriDataSet = "dataset"
5
6 JumlahEpoh = 3
7
8 LabelKelas = (
9     "Kartu Tutup",
10    "Keriting Dua",
11    "Keriting Tiga",
12    "Keriting Empat",
13    "Keriting Lima",
14    "Keriting Enam",
15    "Keriting Tujuh",
16    "Keriting Delapan",
17    "Keriting Sembilan",
18    "Keriting Sepuluh",
19    "Keriting Jack",
20    "Keriting Queen",
21    "Keriting King",
22    "Keriting Ace",
23    "Hati Dua",
24    "Hati Tiga",
25    "Hati Empat",
26    "Hati Lima",
27    "Hati Enam",
28    "Hati Tujuh",
29    "Hati Delapan",
30    "Hati Sembilan",
31    "Hati Sepuluh",
32    "Hati Jack",
33    "Hati Queen",
34    "Hati King",
35    "Hati Ace",
36    "Wajik Dua",
37    "Wajik Tiga",
38    "Wajik Empat",
39    "Wajik Lima",
40    "Wajik Enam",
41    "Wajik Tujuh",
42    "Wajik Delapan",
43    "Wajik Sembilan",
44    "Wajik Sepuluh",
45    "Wajik Jack",
46    "Wajik Queen",
47    "Wajik King",
48    "Wajik Ace",
49    "Sekop Dua",
50    "Sekop Tiga",
51    "Sekop Empat",
52    "Sekop Lima",
53    "Sekop Enam",
54    "Sekop Tujuh",
55    "Sekop Delapan",
56    "Sekop Sembilan",
57    "Sekop Sepuluh",
```

```

58     "Sekop Jack",
59     "Sekop Queen",
60     "Sekop King",
61     "Sekop Ace",
62 )
63
64 # Mulai training
65 # Algoritma Fungsi Training dapat dilihat pada file
# KlasifikasiCNN.py
66 mCNN.TrainingCNN(JumlahEpoh, DirektoriDataSet, LabelKelas, "CardWeight.h5")
67

```

Source Code 3: Training Dataset

3.3 Menu

```

1 import cv2
2
3 def menuState():
4     menu_state = 0
5     howto_state = 1
6     credit_state = 2
7     game_state = 3
8
9     current_state = menu_state
10
11    cap = cv2.VideoCapture(1)
12
13    if not cap.isOpened():
14        print("Cannot open camera")
15        exit()
16
17    font_color = (0,51,25)
18
19    while True:
20        ret, frame = cap.read()
21
22        mainwin = cv2.imread('asset/grass.jpg')
23        mainwin = cv2.resize(mainwin, (853,640))
24
25
26
27        if current_state == menu_state:
28            cv2.putText(mainwin, "Press 'a' to start the game",
29                        (20,280), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
29            cv2.putText(mainwin, "Press 's' to see How to
Play", (20, 320), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
font_color, 3)
30            cv2.putText(mainwin, "Press 'd' to Credit", (20,
360), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
31            cv2.imshow('menu', mainwin)
32
33        key = cv2.waitKey(1)
34        if key == ord('a'):

```

```

35             current_state = game_state
36             break
37         elif key == ord('s'):
38             current_state = howto_state
39         elif key == ord('d'):
40             current_state = credit_state
41
42     elif current_state == howto_state:
43         cv2.putText(mainwin, "- Draw 5 Cards for each
44 player", (20,100), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
45 font_color, 2)
46         cv2.putText(mainwin, "- First round, Put both
47 player card same as dealer card type", (20, 140), cv2.
48 FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
49         cv2.putText(mainwin, "- The player who puts the
50 biggest value wins", (20, 180), cv2.FONT_HERSHEY_SIMPLEX,
51 0.8, font_color, 2)
52         cv2.putText(mainwin, "get the first chance to put
53 a card first", (20, 220), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
54 font_color, 2)
55         cv2.putText(mainwin, "in the next Round ", (20,
56 260), cv2.FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
57         cv2.putText(mainwin, "- Player who got first move
58 can put cards up to player", (20, 300), cv2.
59 FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
60         cv2.putText(mainwin, "the oppenent move after him
61 ", (20, 340), cv2.FONT_HERSHEY_SIMPLEX, 0.8, font_color,
62 2)
63         cv2.putText(mainwin, "Opponent Card have to be
64 the same type", (20, 380), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
65 font_color, 2)
66         cv2.putText(mainwin, "- If player doesnt have a
67 same card type", (20, 420), cv2.FONT_HERSHEY_SIMPLEX,
68 0.8, font_color, 2)
69         cv2.putText(mainwin, "draw a card with showing
70 back side of card", (20, 460), cv2.FONT_HERSHEY_SIMPLEX,
71 0.8, font_color, 2)
72         cv2.putText(mainwin, "- The card player who runs
73 out first wins", (20, 500), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
74 font_color, 2)
75         cv2.putText(mainwin, "Press 'b' to back", (20,
76 600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
77         cv2.imshow('menu', mainwin)
78         key = cv2.waitKey(1)
79         if key == ord('b'):
80             current_state = menu_state
81
82     elif current_state == credit_state:
83         cv2.putText(mainwin, "M. RADHITO BIL ATHO",
84 (20,280), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
85         cv2.putText(mainwin, "COMPUTER ENGINEERING", (20,
86 320), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
87         cv2.putText(mainwin, "INSTITUE OF TECHNOLOGY
88 SEPULUH NOPEMBER", (20, 360), cv2.FONT_HERSHEY_SIMPLEX,
89 0.8, font_color, 3)
90         cv2.putText(mainwin, "Press 'b' to back", (20,

```

```

600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
    cv2.imshow('menu', mainwin)
    key = cv2.waitKey(1)
    if key == ord('b'):
        current_state = menu_state
69
70    if current_state == game_state:
71        break
72
73    cap.release()
74    cv2.destroyAllWindows()
75

```

Source Code 4: Menu

3.4 Close

```

1 import cv2
2
3 def closeState(status):
4
5     cap = cv2.VideoCapture(1)
6
7     if not cap.isOpened():
8         print("Cannot open camera")
9         exit()
10
11    font_color = (255,255,255)
12    close = False
13
14    while True:
15        ret, frame = cap.read()
16
17        mainwin = cv2.imread('asset/grass.jpg')
18        mainwin = cv2.resize(mainwin, (853,640))
19        key = cv2.waitKey(1)
20
21        if status == 1:
22            cv2.putText(mainwin, "P1 Wins !!!", (300, 300),
23            cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
24            cv2.putText(mainwin, "Press 'space' to close the
25            game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
26            font_color, 3)
27            cv2.imshow('close', mainwin)
28            if key == ord(' '):
29                close = True
30                break
31        elif status == 2:
32            cv2.putText(mainwin, "P2 Wins", (300, 300), cv2.
33            FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
34            cv2.putText(mainwin, "Press 'space' to close the
35            game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
36            font_color, 3)
37            cv2.imshow('close', mainwin)
38            if key == ord(' '):
39

```

```

33             close = True
34             break
35         elif status == 3:
36             cv2.putText(mainwin, "DRAW", (300, 300), cv2.
37 FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
38             cv2.putText(mainwin, "Press 'space' to close the
39 game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
40 font_color, 3)
41             cv2.imshow('close', mainwin)
42             if key == ord(' '):
43                 close = True
44                 break
45
46         cap.release()
47         cv2.destroyAllWindows()
48
49

```

Source Code 5: close

3.5 Main

```

1     import numpy as np
2 import cv2
3 import random
4 import KlasifikasiCNN as mCNN
5 import menu
6 import close
7
8 cardName = [
9     "Kartu Tutup",      # 0
10    "Keriting Dua",    # 1
11    "Keriting Tiga",   # 2
12    "Keriting Empat",  # 3
13    "Keriting Lima",  # 4
14    "Keriting Enam",   # 5
15    "Keriting Tujuh",  # 6
16    "Keriting Delapan",# 7
17    "Keriting Sembilan",# 8
18    "Keriting Sepuluh",# 9
19    "Keriting Jack",   # 10
20    "Keriting Queen",  # 11
21    "Keriting King",   # 12
22    "Keriting Ace",    # 13
23    "Hati Dua",        # 14
24    "Hati Tiga",       # 15
25    "Hati Empat",      # 16
26    "Hati Lima",       # 17
27    "Hati Enam",       # 18
28    "Hati Tujuh",      # 19
29    "Hati Delapan",    # 20
30    "Hati Sembilan",   # 21

```

```

31     "Hati Sepuluh",      # 22
32     "Hati Jack",        # 23
33     "Hati Queen",       # 24
34     "Hati King",        # 25
35     "Hati Ace",         # 26
36     "Wajik Dua",        # 27
37     "Wajik Tiga",       # 28
38     "Wajik Empat",      # 29
39     "Wajik Lima",       # 30
40     "Wajik Enam",       # 31
41     "Wajik Tujuh",      # 32
42     "Wajik Delapan",    # 33
43     "Wajik Sembilan",   # 34
44     "Wajik Sepuluh",    # 35
45     "Wajik Jack",       # 36
46     "Wajik Queen",      # 37
47     "Wajik King",       # 38
48     "Wajik Ace",        # 39
49     "Sekop Dua",         # 40
50     "Sekop Tiga",        # 41
51     "Sekop Empat",       # 42
52     "Sekop Lima",        # 43
53     "Sekop Enam",        # 44
54     "Sekop Tujuh",       # 45
55     "Sekop Delapan",     # 46
56     "Sekop Sembilan",   # 47
57     "Sekop Sepuluh",    # 48
58     "Sekop Jack",        # 49
59     "Sekop Queen",       # 50
60     "Sekop King",        # 51
61     "Sekop Ace",         # 52
62 ]
63
64 model = mCNN.LoadModel('CardWeight.h5')
65
66 cap = cv2.VideoCapture(1)
67
68 if not cap.isOpened():
69     print("Cannot open camera")
70     exit()
71
72
73 def binerimg(img):
74     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
75     blur = cv2.GaussianBlur(gray, (5, 5), 0)
76     canny = cv2.Canny(blur, 100, 150)
77     kernel = np.ones((3, 3))
78     dial = cv2.dilate(canny, kernel=kernel, iterations=1)
79     imgThres = cv2.erode(dial, kernel, iterations=1)
80     return canny
81
82 def getContours(img, kontur, draw = False):
83
84     contours, hier = cv2.findContours(img, cv2.RETR_EXTERNAL,
85                                         cv2.CHAIN_APPROX_NONE)

```

```

86     cropped_frames = []
87     cardCoor = []
88
89     for cnt in contours:
90         area = cv2.contourArea(cnt)
91         perimeter = cv2.arcLength(cnt, True)
92
93         if area > 5000:
94             approx = cv2.approxPolyDP(cnt, 0.02 * perimeter,
95             True)
96             numCorners = len(approx)
97
98             if numCorners == 4:
99                 x,y,w,h = cv2.boundingRect(approx)
100                cardCoor.append((x,y))
101
102                if draw:
103                    cv2.rectangle(kontur, (x,y), (x+w, y+h),
104 (0,255,0), 3)
105
106                cropped_frame = img[y:y+h, x:x+w]
107
108                cropped_frames.append(cropped_frame)
109
110                for i, cropped_frame in enumerate(cropped_frames):
111                    cv2.namedWindow(f'Cropped Frame{i}', cv2.
112 WINDOW_NORMAL)
113                    cv2.resizeWindow(f'Cropped Frame{i}', cropped_frame.
114 shape[1], cropped_frame.shape[0])
115                    # cv2.imshow(f'Cropped Frame{i}', cropped_frame)
116
117                return cropped_frames, cardCoor
118
119
120    def cardType(card_index):
121        if 1<= card_index <= 13:
122            return "Keriting"
123        elif 14<= card_index <= 26:
124            return "Hati"
125        elif 27<= card_index <= 39:
126            return "Wajik"
127        elif 40<= card_index <= 52:
128            return "Sekop"
129        else:
130            return None
131
132
133    def comparedCard(card1, card2):
134
135        value1 = cardName.index(card1)
136        value2 = cardName.index(card2)
137
138        type1 = cardType(value1)
139        type2 = cardType(value2)
140
141        if type1 is not None and type1 == type2:
142            if value1 > value2:
143                return 1

```

```

138         elif value1 < value2:
139             return -1
140         else:
141             return 0
142     else:
143         return None
144
145 def predik(bgr_frames, cardCoor):
146     storedCard = []
147
148     for i, bgr_frame in enumerate(bgr_frames):
149         X = []
150
151         img = cv2.resize(bgr_frame, (128, 128))
152         img = img.astype('float32') / 255 # Normalize to [0,
153                                         1]
154         X.append(img)
155
156         X = np.array(X)
157         X = X.astype('float32')
158
159         hs = model.predict(X, verbose = 0)
160         n = np.max(np.where(hs== hs.max()))
161
162
163         cv2.putText(CardRes_bg, f'{cardName[n]}{{:.2f}}'.
164         format(hs[0,n]), cardCoor, cv2.FONT_HERSHEY_SIMPLEX,
165         0.5, font_color, 1)
166         storedCard.append(cardName[n])
167
168     return storedCard
169
170     debounce_timer = 0
171
172 def decider(storedCard1, storedCard2):
173     global turn
174     global P1_Cardleft
175     global P2_Cardleft
176     global debounce_timer
177
178     if len(storedCard1) ==1 and len(storedCard2) == 1:
179         winner = comparedCard(storedCard1[0], storedCard2[0])
180         if winner == 1:
181             cv2.putText(frame, f'{storedCard1[0]} wins!', (10, 20),
182             cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)
183             turn = 1
184             if debounce_timer <= 0:
185                 P1_Cardleft = P1_Cardleft - 1
186                 P2_Cardleft = P2_Cardleft - 1
187                 debounce_timer = 200
188             else:
189                 debounce_timer -= 1
190
191         elif winner == -1:
192             cv2.putText(frame, f'{storedCard2[0]} wins!', (10, 20),
193             cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)
194             turn = 2
195             if debounce_timer <= 0:
196                 P1_Cardleft = P1_Cardleft - 1
197                 P2_Cardleft = P2_Cardleft - 1
198                 debounce_timer = 200
199             else:
200                 debounce_timer -= 1
201
202     return turn, P1_Cardleft, P2_Cardleft, debounce_timer

```

```

(10, 20), cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)
    turn = 2
    if debounce_timer <= 0:
        P1_Cardleft = P1_Cardleft - 1
        P2_Cardleft = P2_Cardleft - 1
        debounce_timer = 200
    else:
        debounce_timer -= 1

    else:
        cv2.putText(stat_bg, "Card not Eligible", (10,
20), cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)
        print("Card not eligible")
        print(comparedCard(storedCard1[0], storedCard2
[0]))

def border(frame):
    frame = cv2.copyMakeBorder(frame, 3,3,3,3, cv2.
BORDER_CONSTANT, (0,0,0))
    return frame

font_color = (255, 255, 255) # White color in BGR format

cards_without_tutup = [card for card in cardName if card != "Kartu Tutup"]
randomCard = random.choice(cards_without_tutup)
indexRandomCard = cardName.index(randomCard)
typeRandomCard = cardType(indexRandomCard)

turn = 0
P1_Cardleft = 5
P2_Cardleft = 5

Menu_state = True
Closing_state = 0
status = 0

while True:
    # Capture frame-by-frame
    if Menu_state:
        menu.menuState()
        Menu_state = False

    ret, frame = cap.read()
    frame = border(frame)

    mainwin = cv2.imread('asset/grass.jpg')
    mainwin = cv2.resize(mainwin, (853,640))

    # DisScore_bg = np.ones((100,193,3), dtype=np.uint8)
    DisScore_bg = cv2.imread('asset/dirt.jpeg')
    DisScore_bg = cv2.resize(DisScore_bg, (187,154))
    DisScore_bg = border(DisScore_bg)

```

```

241     cv2.putText(DisScore_bg, "Card Left", (48, 25), cv2.
242 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
243     cv2.putText(DisScore_bg, "P1", (30, 69), cv2.
244 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
245     cv2.putText(DisScore_bg, "P2", (130, 69), cv2.
246 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
247     cv2.putText(DisScore_bg, f"{P1_Cardleft}", (20, 140), cv2.
248 FONT_HERSHEY_SIMPLEX, 2.5, font_color, 1)
249     cv2.putText(DisScore_bg, f"{P2_Cardleft}", (120, 140),
250 cv2.FONT_HERSHEY_SIMPLEX, 2.5, font_color, 1)

251 CardRes_bg = cv2.imread('asset/dirt.jpeg')
252 CardRes_bg = cv2.resize(CardRes_bg, (629,124))
253 CardRes_bg = border(CardRes_bg)
254 cv2.putText(CardRes_bg, "P1 Card", (110, 30), cv2.
255 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
256 cv2.putText(CardRes_bg, "P2 Card", (440, 30), cv2.
257 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)

258 stat_bg = cv2.imread('asset/dirt.jpeg')
259 stat_bg = cv2.resize(stat_bg, (187,284))
260 stat_bg = border(stat_bg)

261 biner = binerimg(frame)
262 kontur = frame.copy()
263 height, width, _ = frame.shape
264 midPoint = width // 2
265 cv2.line(frame, (midPoint,0), (midPoint, height),
266 (0,0,255), 5)

267 p1side = biner[:, :midPoint]
268 p2side = biner[:, midPoint:]
269 # cv2.imshow('p1side', p1side)
270 # cv2.imshow('p2side', p2side)

271 cropped_frames1, cardCoor1 = getContours(p1side, kontur,
272 draw=True)
273 cropped_frames2, cardCoor2 = getContours(p2side, kontur,
274 draw=True)

275 bgr_frames1 = [cv2.cvtColor(cropped_frame, cv2.
276 COLOR_GRAY2BGR) for cropped_frame in cropped_frames1]
277 bgr_frames2 = [cv2.cvtColor(cropped_frame, cv2.
278 COLOR_GRAY2BGR) for cropped_frame in cropped_frames2]
279 # for i, bgr_frame in enumerate(bgr_frames1):
280 #     cv2.namedWindow(f'BGR Frame {i}', cv2.WINDOW_NORMAL)
281 #     cv2.resizeWindow(f'BGR Frame {i}', bgr_frame.shape
282 [1], bgr_frame.shape[0])
283 #     cv2.imshow(f'BGR Frame {i}', bgr_frame)

284 # cv2.imshow('biner', biner)

285 # cv2.imshow('contur', kontur)

286 font_color = (255, 255, 255) # White color in BGR format

```

```

284     storedCard1 = predik(bgr_frames1, (60, 60))
285     storedCard2 = predik(bgr_frames2, (370, 60))
286
287     ## POSISI TURN
288     if turn == 1:
289         print("Player 1 Move First")
290         cv2.putText(stat_bg, "Player 1 Move First", (10, 20),
291                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
292         if storedCard2 and not storedCard1:
293             cv2.putText(stat_bg, "Its not your turn", (10,
294 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
295         else :
296             decider(storedCard1, storedCard2)
297     elif turn == 2:
298         cv2.putText(stat_bg, "Player 2 Move First", (10, 20),
299                     cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
300         if storedCard1 and not storedCard2:
301             cv2.putText(stat_bg, "Its not your turn", (10,
302 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
303         else :
304             decider(storedCard1, storedCard2)
305     elif turn == 0:
306         cv2.putText(frame, "Dealer Card", (253, 50), cv2.
307 FONT_HERSHEY_SIMPLEX, 1, font_color, 1, cv2.LINE_4)
308         cv2.putText(frame, f"{randomCard}", (255, 80), cv2.
309 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1, cv2.LINE_4)
310         if storedCard1 and storedCard2:
311             iCardP1 = cardName.index(storedCard1[0])
312             # print("value 1 =", value1)
313             iCardP2 = cardName.index(storedCard2[0])
314             # print("value 2 =", value2)
315
316             type1 = cardType(iCardP1)
317             # print("type 1= "+ type1)
318             type2 = cardType(iCardP2)
319             # print("type 2= " + type2)
320             if iCardP1 or iCardP2 == indexRandomCard:
321                 print("That card is a Dealer card")
322                 print("Please Draw Another Card")
323
324             if type1 == type2 == typeRandomCard:
325                 if iCardP1 > iCardP2:
326                     print("Player 1 Win")
327                     cv2.putText(stat_bg, "Player 1 Win", (10,
328 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
329                     cv2.putText(frame, "Player 1 Win", (230,
330 400), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
331                     turn = 1
332                     if debounce_timer <= 0:
333                         P1_Cardleft = P1_Cardleft - 1
334                         P2_Cardleft = P2_Cardleft - 1
335                         debounce_timer = 200
336                     else:
337                         debounce_timer -= 1
338
339             elif iCardP1 < iCardP2:

```

```

332             print("Player 2 Win")
333             cv2.putText(stat_bg, "Player 2 Win", (10,
334     270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
335             cv2.putText(frame, "Player 2 Win", (230,
336     400), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
337             turn = 2
338             if debounce_timer <= 0:
339                 P1_Cardleft = P1_Cardleft - 1
340                 P2_Cardleft = P2_Cardleft - 1
341                 debounce_timer = 200
342             else:
343                 debounce_timer -= 1
344             else:
345                 print("Something Wrong")
346             else:
347                 print("Card not Eligible")
348
349             if storedCard1:
350                 if storedCard1[0]== "Kartu Tutup":
351                     print("Player 1, please draw a Card")
352                     cv2.putText(stat_bg, "Player 1, please draw a
353     Card", (10, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
354     font_color, 1)
355                     if debounce_timer <= 0:
356                         P1_Cardleft = P1_Cardleft + 1
357                         debounce_timer = 200
358                     else:
359                         debounce_timer -= 1
360                     elif storedCard2:
361                         if storedCard2[0]== "Kartu Tutup":
362                             print("Player 2, please draw a Card")
363                             cv2.putText(stat_bg, "Player 2, please draw a
364     Card", (10, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
365     font_color, 1)
366                             if debounce_timer <= 0:
367                                 P2_Cardleft = P2_Cardleft + 1
368                                 debounce_timer = 200
369                             else:
370                                 debounce_timer -= 1
371
372             if P1_Cardleft == 0:
373                 print("Player 1 Win!")
374                 Closing_state = 1
375                 status = 1
376             elif P2_Cardleft == 0:
377                 print("Player 2 Win!")
378                 Closing_state = 1
379                 status = 2
380             elif P1_Cardleft == 0 and P2_Cardleft == 0:
381                 print("Draw")
382                 Closing_state = 1
383                 status = 3
384
385             if debounce_timer > 0:
386                 debounce_timer -= 1

```

```

382
383     cv2.putText(stat_bg, f"scan after = {debounce_timer}", (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
384     cv2.imshow('asli', frame)
385     mainwin[0 : frame.shape[0], 0 : frame.shape[1]] = frame
386     mainwin[10: 170, 650: 843] = DisScore_bg
387     mainwin[490: 620, 5: 640] = CardRes_bg
388     mainwin[190: 480, 650: 843] = stat_bg
389     cv2.imshow("Main", mainwin)
390
391     print(f'timer = {debounce_timer}')
392
393     # print(f'P1_Cardleft = {P1_Cardleft} & P2_Cardleft = {P2_Cardleft}')
394
395     if Closing_state == 1:
396         close.closeState(status)
397         Closing_state = 2
398
399     if (cv2.waitKey(1) == ord('q')) or (Closing_state == 2):
400         break
401 # When everything done, release the capture
402 cap.release()
403 cv2.destroyAllWindows()
404

```

Source Code 6: Main

4 Pembahasan

4.1 Labelling

```
1   cardName = [
2     "Kartu Tutup",
3     "Keriting Dua",
4     "Keriting Tiga",
5     "Keriting Empat",
6     "Keriting Lima",
7     "Keriting Enam",
8     "Keriting Tujuh",
9     "Keriting Delapan",
10    "Keriting Sembilan",
11    "Keriting Sepuluh",
12    "Keriting Jack",
13    "Keriting Queen",
14    "Keriting King",
15    "Keriting Ace",
16    "Hati Dua",
17    "Hati Tiga",
18    "Hati Empat",
19    "Hati Lima",
20    "Hati Enam",
21    "Hati Tujuh",
22    "Hati Delapan",
23    "Hati Sembilan",
24    "Hati Sepuluh",
25    "Hati Jack",
26    "Hati Queen",
27    "Hati King",
28    "Hati Ace",
29    "Wajik Dua",
30    "Wajik Tiga",
31    "Wajik Empat",
32    "Wajik Lima",
33    "Wajik Enam",
34    "Wajik Tujuh",
35    "Wajik Delapan",
36    "Wajik Sembilan",
37    "Wajik Sepuluh",
38    "Wajik Jack",
39    "Wajik Queen",
40    "Wajik King",
41    "Wajik Ace",
42    "Sekop Dua",
43    "Sekop Tiga",
44    "Sekop Empat",
45    "Sekop Lima",
46    "Sekop Enam",
47    "Sekop Tujuh",
48    "Sekop Delapan",
49    "Sekop Sembilan",
50    "Sekop Sepuluh",
51    "Sekop Jack",
```

```

52     "Sekop Queen",
53     "Sekop King",
54     "Sekop Ace",
55 ]
56

```

Source Code 7: Label

Bagian diatas merupakan bagian labelling yang digunakan pada setiap file. Hal ini digunakan untuk memberikan nama atau label pada setiap kartu yang dideteksi dan mengurutkannya.

4.2 Build Dataset

```

1   import cv2
2   import os
3   import datetime
4   import time
5   import numpy as np
6

```

Source Code 8: Library

Library diatas digunakan untuk pengolahan citra, akses folder dari komputer, akses tanggal, akses perhitungan waktu, dan perhitungan dalam python.

```

1 # Fungsi penamaan file berdasarkan waktu pengambilan
2 def GetFileName():
3     x = datetime.datetime.now()
4     s = x.strftime(' %Y-%m-%d-%H%M%S%f ')
5     return s
6

```

Source Code 9: Penamaan file

fungsi diatas digunakan untuk memberikan nama file yang telah dicapture berdasarkan waktu pengambilannya.

```

1 # Fungsi menentukan luas area dari 4 titik
2 def polygon_area(points):
3     points = np.vstack((points, points[0]))
4     area = 0.5 * np.abs(np.dot(points[:, 0], np.roll(
5         points[:, 1], 1)) - np.dot(points[:, 1], np.roll(points[:, 0], 1)))
6     return area

```

Source Code 10: area titik

Fungsi diatas untuk menghitung luas daerah dari 4 titik yang telah ditemukan. Perhitungan dengan cara Shoelace formula dengan menghitung selisih produk dot dari koordinat x dan y titik-titik poligon.

```

1 def CreateDir(path):
2     ls = []
3     head_tail = os.path.split(path)
4     ls.append(path)
5     while len(head_tail[1])>0:

```

```

6         head_tail = os.path.split(path)
7         path = head_tail[0]
8         ls.append(path)
9         head_tail = os.path.split(path)
10        for i in range(len(ls)-2,-1,-1):
11            sf = ls[i]
12            isExist = os.path.exists(sf)
13            if not isExist:
14                os.makedirs(sf)
15

```

Source Code 11: Create Directory

Fungsi diatas adalah untuk membuat direktori tempat yang akan menyimpan gambar output. Dilakukan pengecekan tiap bagian dan membuat direktori jika belum ada. Pengecekan tersebut dilakukan dengan memecah path menjadi parents and childern dan menyimpan parents tersebut dalam variable head_tail lalu melakukan loop selama tail ada dan mengupdate pathnya. Pada loop kedualah yang akan membuat folder direktori baru jika belum ada.

```

1 def CreateDataSet(sDirektoriData,sKelas,NoKamera,
2   FrameRate):
3     global cardName, cardNameIndex
4
5     cap = cv2.VideoCapture(NoKamera)
6     TimeStart = time.time()
7
8     # Mengetahui limit detik dalam pengambilan gambar
9     saveTimeLimit = time.time()
10
11    # Status record
12    isSaving = False
13
14    while cap.isOpened():
15        success, frame = cap.read()
16
17        # Membuat direktori dataset sesuai label
18        sDirektoriKelas = sDirektoriData+"/"+cardName[
19          cardNameIndex]
20        CreateDir(sDirektoriKelas)
21
22        if not success:
23            print("Ignoring empty camera frame.")
24            continue
25
26        isDetected = False
27
28        # Image Processing untuk filter citra dengan edge
29        # detection
30        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
31        blur = cv2.GaussianBlur(gray, (5, 5), 0)
32        canny = cv2.Canny(blur, 100, 150)
33        cv2.imshow("2. Threshold", canny)
34
35
36        # Mengambil data dari komponen yang terhubung atau

```

```

        garis dengan ukuran yang diinginkan
34         totalLabels, label_ids, values, centroid = cv2.
connectedComponentsWithStats(canny, 4, cv2.CV_32S)
35         bigIndex = []
36         for i in range(totalLabels):
37             hw = values[i,2:4]
38             if (70<hw[0]<200 and 150<hw[1]<300):
39                 bigIndex.append(i)
40
41         # Dari garis yang telah diketahui, diberi gambar
42         persegi (contour)
43         for i in bigIndex:
44             topLeft = values[i,0:2]
45             bottomRight = values[i,0:2]+values[i,2:4]
46             frame = cv2.rectangle(frame, topLeft,
bottomRight, color=(0,0,255), thickness=3)
47             # Disini ada break, yg berarti kita cuma
48             ngambil 1 item doang
49             break
50         cv2.imshow("4. Hasil habis dikotakin", frame)
51
52         # Gambar yang berada dalam contour akan diambil (
53         crop)
54         for i in bigIndex:
55             topLeft = values[i,0:2]
56             bottomRight = values[i,0:2]+values[i,2:4]
57             cardImage = canny[topLeft[1]:bottomRight[1],
topLeft[0]:bottomRight[0]]
58             cv2.imshow('5. Hasil dari cardImage',
cardImage)
59             break
60
61         # Simpan gambar yang dicrop sesuai berjalannya
62         waktu
63         TimeNow = time.time()
64         if TimeNow-TimeStart>1/FrameRate:
65             sfFile = sDirektoriKelas+"/"+GetFileName()
66             if isSaving and len(bigIndex) > 0:
67                 cv2.imwrite(sfFile+'.jpg', cardImage)
68             TimeStart = TimeNow
69
70             cv2.putText(frame, "Nama Kartu yg direkam:", (0,
71             30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
72             cv2.putText(frame, f"{cardNameIndex+1}. " +
73             cardName[cardNameIndex], (0, 70), cv2.FONT_HERSHEY_SIMPLEX,
0.8, (0, 0, 255), 2)
74             cv2.putText(frame, "Tekan spasi untuk mulai
record", (0, 400), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,
0,0), 2)
75
76             if isSaving:
77                 cv2.putText(frame, "Record", (0, 30), cv2.
FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
78             else:
79                 saveTimeLimit = time.time()

```

```

75
76         cv2.imshow("Tampilan akhir", frame)
77
78         key = cv2.waitKey(5)
79
80         # Tekan spasi untuk mulai rekam/menyimpan gambar
81         if key == 32:
82             isSaving = not isSaving
83
84             # Kalo udah lebih dari 8 detik, record/
penyimpanan foto selesai
85             if time.time() - saveTimeLimit >= 8:
86                 cardNameIndex += 1
87                 isSaving = False
88
89             if key & 0xFF == 27:
90                 break
91             cap.release()
92             cv2.destroyAllWindows()
93

```

Source Code 12: CreateDataSet

Fungsi diatas adalah fungsi utama dari pembuatan direktori hasil tangkap gambar dan memanfaatkan fungsi sebelumnya. Dengan melakukan filter pada citra/frame, dapat dilakukan deteksi contour untuk membatasi atau menangkap bentuk dari kartu. Kartu tersebut akan dicrop akan direkam / dicapture setiap berjalaninya waktu. Pengambilan kartu diharapkan untuk digerakkan untuk mendapatkan variasi pose guna meningkatkan akurasi prediksi.

4.3 Klasifikasi CNN

```

1   import os
2   from keras.models import load_model
3   import cv2
4   import numpy as np
5   from keras.layers import Input, Dense
6   from keras.layers import Conv2D, MaxPooling2D, Flatten
7   from keras.models import Model
8   import matplotlib.pyplot as plt
9   from datetime import datetime
10  from numpy import expand_dims
11  from keras.utils import load_img
12  from keras.utils import img_to_array
13  from keras.preprocessing.image import ImageDataGenerator
14  from matplotlib import pyplot
15

```

Source Code 13: Library

Library diatas digunakan untuk image processing, perhitungan python, keras untuk machine learning dengan mmdel CNN, dan untuk visualisasinya.

```

1   def LoadCitraTraining(sDir, LabelKelas):
2       JumlahKelas=len(LabelKelas)

```

```

3     TargetKelas = np.eye(JumlahKelas)
4
5     X=[] #Menampung Data Citra
6     T=[] #Menampung Target
7     for i in range(len(LabelKelas)):
8         #Membaca file citra di setiap direktori data set
9         DirKelas = os.path.join(sDir, LabelKelas[i])
10        files = os.listdir(DirKelas)
11        for f in files:
12            ff=f.lower()
13            print(f)
14            #memilih citra dengan extensi jpg,jpeg,dan png
15            if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.
endswith('.png')):
16                NmFile = os.path.join(DirKelas,f)
17                img= np.double(cv2.imread(NmFile,1))
18                img=cv2.resize(img,(128,128))
19                img= np.asarray(img)/255
20                img=img.astype('float32')
21
22                X.append(img)
23                T.append(TargetKelas[i])
24
25    #Mengubah List Menjadi numpy array
26    X=np.array(X)
27    T=np.array(T)
28    X=X.astype('float32')
29    T=T.astype('float32')
30    return X,T
31

```

Source Code 14: Load Citra

Fungsi diatas digunakan untuk mengambil gambar hasil dari Build Dataset dengan mempertimbangkan banyaknya kelas dan melihat target dari kelas tersebut. Dilakukan filter untuk mengambil data jpg/jpeg/png dan dilakukan normalisasi dari format gambar dan ukuran gambar. Gambar - gambar tersebut akan disimpan dalam array.

```

1   def ModelDeepLearningCNN(JumlahKelas):
2       input_img = Input(shape=(128, 128, 3))
3       x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)
4       x = MaxPooling2D((2, 2), padding='same')(x)
5       x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
6       x = MaxPooling2D((2, 2), padding='same')(x)
7       x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
8       x = Flatten()(x)
9       x = Dense(100,activation='relu')(x)
10      x=Dense(JumlahKelas,activation='softmax')(x)
11      ModelCNN = Model(input_img, x)
12      ModelCNN.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
13      return ModelCNN

```

Source Code 15: Model CNN

Fungsi diatas adalah penyusunan layer untuk model CNN. Model disusun sesuai pada Figure 1, yaitu input layer, Convolutional Layer, Pooling Layer, Fully Connected Layer, dan Output Layer. Convolutional layer dilakukan 3 lapis dan Pooling layer 2 lapis.

```
1 def TrainingCNN(JumlahEpoh,DirektoriDataSet,LabelKelas,  
2 NamaFileBobot):  
3     #Membaca Data training dan label Kelas  
4     X,D=LoadCitraTraining(DirektoriDataSet,LabelKelas)  
5     JumlahKelas = len(LabelKelas)  
6     #Membuat Model CNN  
7     ModelCNN =ModelDeepLearningCNN(JumlahKelas)  
8     #Trainng  
9     history=ModelCNN.fit(X, D,epochs=JumlahEpoh,shuffle=  
10    True)  
11    #Menyimpan hasil learning  
12    ModelCNN.save(NamaFileBobot)  
13  
14  
15        return ModelCNN,history  
16    )
```

Source Code 16: TrainingDataset

Fungsi diatas untuk training dari gambar yang telah diambil dan di proses. Training dilakukan sesuai urutan pada array dan membuat model sebanyak kelas yang dibutuhkan. Training dilakukan sesuai banyaknya epoch yang diinput dan diubah kedalam nama file weight, yaitu h5

```
1 def Klasifikasi(DirDataSet ,DirKlasifikasi ,LabelKelas ,
2 ModelCNN=[]):
3     # untuk menampung input
4     X=[]
5
6     ls = []
7     DirKelas = DirDataSet+"/"+DirKlasifikasi
8     print(DirKelas)
9     files = os.listdir(DirKelas)
10    n=0
11    for f in files:
12        ff=f.lower()
13        print(f)
14        if (ff.endswith('.jpg')|ff.endswith('.jpeg')|ff.
15 endswith('.png')):
16            ls.append(ff)
17            NmFile = os.path.join(DirKelas,f)
18            img= cv2.imread(NmFile ,1)
19            img=cv2.resize(img ,(128 ,128))
20            img= np.asarray(img)/255
21            img=img.astype('float32')
22            X.append(img)
23
24    X=np.array(X)
```

```

23     X=X.astype('float32')
24
25     #prediksi hasil klasifikasi
26     hs=ModelCNN.predict(X)
27
28     LKlasifikasi=[]
29     LKelasCitra =[]
30     n = X.shape[0]
31     for i in range(n):
32         v=hs[i,:]
33         if v.max()>0.5:
34             idx = np.max(np.where( v == v.max()))
35             LKelasCitra.append(LabelKelas[idx])
36         else:
37             idx=-1
38             LKelasCitra.append(" - ")
39             LKlasifikasi.append(idx)
40     LKlasifikasi = np.array(LKlasifikasi)
41     return ls, hs, LKelasCitra
42

```

Source Code 17: Klasifikasi

Fungsi diatas adalah untuk pengklasifikasian citra dari hasil gambar yang telah diload pada LoadCitra. Gambar - gambar tersebut dimasukkan ke dalam satu array untuk satu kelas bergantian. Gambar akan diprediksi dari hasil training yang telah dilakukan. Dengan hasil training tersebut, dapat dilakukan prediksi pada array yang telah dibuat. Hasil dari prediksi tersebut akan didapatkan score hasil prediksi dan kelas yang sekiranya sesuai berdasarkan score yang didapat.

4.4 Training Data

```

1   import KlasifikasiCNN as mCNN
2   DirektoriDataSet = "dataset"
3   JumlahEpoh = 3
4
5   LabelKelas = (
6   .
7   .
8   )
9
10  mCNN.TrainingCNN(JumlahEpoh, DirektoriDataSet, LabelKelas,
11      , "CardWeight.h5")
12

```

Source Code 18: Pengaplikasian Training data

Fungsi diatas adalah pengaplikasian dari fungsi training KlasifikasiCNN dengan jumlah epoch 3 dan membuatnya menjadi file bobot yaitu CardWeight.h5

4.5 Menu

Pada file ini hanya dibutuhkan library OpenCV dikarenakan untuk pembuatan menu dan difokuskan pada visualisasi menu tersebut. Algoritmanya sendiri dapat dilakukan hanya dengan python.

```
1 menu_state = 0
2 howto_state = 1
3 credit_state = 2
4 game_state = 3
5
6 current_state = menu_state
7
8 font_color = (0,51,25)
9
```

Source Code 19: initialization

Dilakukan inisialisasi untuk menentukan posisi state dari menu, yaitu menu, howto, credit, dan game state. variable current_state digunakan untuk mengetahui kondisi state saat ini. Inisialisasi juga dilakukan pada warna font yang digunakan.

```
1 def menuState():
2     menu_state = 0
3     howto_state = 1
4     credit_state = 2
5     game_state = 3
6
7     current_state = menu_state
8
9     font_color = (0,51,25)
10
11    while True:
12
13
14        mainwin = cv2.imread('asset/grass.jpg')
15        mainwin = cv2.resize(mainwin, (853,640))
16
17
18        if current_state == menu_state:
19            cv2.putText(mainwin, "Press 'a' to start the game",
20                        (20,280), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
21            cv2.putText(mainwin, "Press 's' to see How to
22 Play", (20, 320), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
23            font_color, 3)
24            cv2.putText(mainwin, "Press 'd' to Credit", (20,
25            360), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
26            cv2.imshow('menu', mainwin)
27
28        key = cv2.waitKey(1)
29        if key == ord('a'):
30            current_state = game_state
31            break
32        elif key == ord('s'):
33            current_state = howto_state
```

```

31         elif key == ord('d'):
32             current_state = credit_state
33
34         elif current_state == howto_state:
35             cv2.putText(mainwin, "- Draw 5 Cards for each
36             player", (20,100), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
37             font_color, 2)
38             cv2.putText(mainwin, "- First round, Put both
39             player card same as dealer card type", (20, 140), cv2.
40             FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
41             cv2.putText(mainwin, "- The player who puts the
42             biggest value wins", (20, 180), cv2.FONT_HERSHEY_SIMPLEX,
43             0.8, font_color, 2)
44             cv2.putText(mainwin, "get the first chance to put
45             a card first", (20, 220), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
46             font_color, 2)
47             cv2.putText(mainwin, "in the next Round ", (20,
48             260), cv2.FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
49             cv2.putText(mainwin, "- Player who got first move
50             can put cards up to player", (20, 300), cv2.
51             FONT_HERSHEY_SIMPLEX, 0.8, font_color, 2)
52             cv2.putText(mainwin, "the oppenent move after him
53             ", (20, 340), cv2.FONT_HERSHEY_SIMPLEX, 0.8, font_color,
54             2)
55             cv2.putText(mainwin, "Opponent Card have to be
56             the same type", (20, 380), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
57             font_color, 2)
58             cv2.putText(mainwin, "- If player doesnt have a
59             same card type", (20, 420), cv2.FONT_HERSHEY_SIMPLEX,
60             0.8, font_color, 2)
61             cv2.putText(mainwin, "draw a card with showing
62             back side of card", (20, 460), cv2.FONT_HERSHEY_SIMPLEX,
63             0.8, font_color, 2)
64             cv2.putText(mainwin, "- The card player who runs
65             out first wins", (20, 500), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
66             font_color, 2)
67             cv2.putText(mainwin, "Press 'b' to back", (20,
68             600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
69             cv2.imshow('menu', mainwin)
70             key = cv2.waitKey(1)
71             if key == ord('b'):
72                 current_state = menu_state
73
74             elif current_state == credit_state:
75                 cv2.putText(mainwin, "M. RADHITO BIL ATHO",
76                 (20,280), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
77                 cv2.putText(mainwin, "COMPUTER ENGINEERING", (20,
78                 320), cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
79                 cv2.putText(mainwin, "INSTITUE OF TECHNOLOGY
80                 SEPULUH NOPEMBER", (20, 360), cv2.FONT_HERSHEY_SIMPLEX,
81                 0.8, font_color, 3)
82                 cv2.putText(mainwin, "Press 'b' to back", (20,
83                 600), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,0), 2)
84                 cv2.imshow('menu', mainwin)
85                 key = cv2.waitKey(1)
86                 if key == ord('b'):

```

```

60         current_state = menu_state
61
62     if current_state == game_state:
63         break
64
65     cv2.destroyAllWindows()
66

```

Source Code 20: Menu

Diatas digunakan untuk menjalankan menu state. Dalam Menu state ini, memiliki 3 arah, yaitu untuk memulai game, mengakses halaman how to play, dan mengakses halaman credit. Pada halaman how to play berisi petunjuk dan aturan dalam permainan. Dengan memulai game, akan menutup window dan memulai ke window utama yaitu window game tersebut.

```

1 def closeState(status):
2
3     font_color = (255, 255, 255)
4     close = False
5
6     while True:
7
8         mainwin = cv2.imread('asset/grass.jpg')
9         mainwin = cv2.resize(mainwin, (853, 640))
10        key = cv2.waitKey(1)
11
12        if status == 1:
13            cv2.putText(mainwin, "P1 Wins !!!", (300, 300),
14            cv2.FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
15            cv2.putText(mainwin, "Press 'space' to close the
16            game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
17            font_color, 3)
18            cv2.imshow('close', mainwin)
19            if key == ord(' '):
20                close = True
21                break
22        elif status == 2:
23            cv2.putText(mainwin, "P2 Wins", (300, 300), cv2.
24            FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
25            cv2.putText(mainwin, "Press 'space' to close the
26            game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
27            font_color, 3)
28            cv2.imshow('close', mainwin)
29            if key == ord(' '):
30                close = True
31                break
32        elif status == 3:
33            cv2.putText(mainwin, "DRAW", (300, 300), cv2.
34            FONT_HERSHEY_SIMPLEX, 1.5, font_color, 3)
35            cv2.putText(mainwin, "Press 'space' to close the
36            game", (30, 340), cv2.FONT_HERSHEY_SIMPLEX, 1.5,
37            font_color, 3)
38            cv2.imshow('close', mainwin)
39            if key == ord(' '):
40                close = True

```

```

32         break
33
34     if close:
35         break
36
37 cv2.destroyAllWindows()
38

```

Source Code 21: close

Dalam close state ini adalah halaman/window diaman telah ditentukan pemenangnya dan transisi untuk menutup game.

4.6 Main

```

1 import numpy as np
2 import cv2
3 import random
4 import KlasifikasiCNN as mCNN
5 import menu
6 import close
7
8 model = mCNN.LoadModel('CardWeight.h5')
9
10 cap = cv2.VideoCapture(1)
11
12 if not cap.isOpened():
13     print("Cannot open camera")
14     exit()
15

```

Source Code 22: Library

Pada library ini, akan meng-import file - file yang telah dibuat yang berisi fungsi - fungsi penunjang berjalannya game dan deteksi. Dengan library tersebut, dapat kita persiapkan camera dan bobot dari training.

```

1 def binerimg(img):
2     gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
3     blur = cv2.GaussianBlur(gray, (5, 5), 0)
4     canny = cv2.Canny(blur, 100, 150)
5     return canny
6

```

Source Code 23: Image Preprocessing

Fungsi diatas untuk image preprocessing atau filterisasi agar image mudah dideteksi

```

1 def getContours(img, kontur, draw = False):
2
3     contours, hier = cv2.findContours(img, cv2.RETR_EXTERNAL,
4                                         cv2.CHAIN_APPROX_NONE)
5
6     cropped_frames = []
7     cardCoor = []
8

```

```

8     for cnt in contours:
9         area = cv2.contourArea(cnt)
10        perimeter = cv2.arcLength(cnt, True)
11
12        if area > 5000:
13            approx = cv2.approxPolyDP(cnt, 0.02 * perimeter,
14            True)
14            numCorners = len(approx)
15
16            if numCorners == 4:
17                x,y,w,h = cv2.boundingRect(approx)
18                cardCoor.append((x,y))
19
20            if draw:
21                cv2.rectangle(kontur, (x,y), (x+w, y+h),
21 (0,255,0), 3)
22
23            cropped_frame = img[y:y+h, x:x+w]
24
25            cropped_frames.append(cropped_frame)
26
27        for i, cropped_frame in enumerate(cropped_frames):
28            cv2.namedWindow(f'Cropped Frame{i}', cv2.
28 WINDOW_NORMAL)
29            cv2.resizeWindow(f'Cropped Frame{i}', cropped_frame.
30 shape[1], cropped_frame.shape[0])
30            # cv2.imshow(f'Cropped Frame{i}', cropped_frame)
31
32    return cropped_frames, cardCoor
33

```

Source Code 24: Contour

Fungsi diatas adalah cara lain untuk mendeteksi contour dari kartu dengan memanfaatkan library dari opencv. Tahap pertama dilakukan pencarian contour dari image. Pencarian tersebut dipermudah dengan hasil image pre-processing dengan hanya mendeteksi satu unsur warna yang terhubung dan kontras. Setelah contour ditemukan, inisialisasi area dan panjang dari contour untuk mengetahui atau memilih kartu yang diberi contour. Hal itu dilakukan dengan menentukan toleransi ujung contour, memastikan hanya memiliki 4 ujung contour. Dari hal tersebut, digambar bentuk contournya dan diketahui sudut-sudutnya untuk mengetahui coordinat dari contour.

```

1 def cardType(card_index):
2     if 1<= card_index <= 13:
3         return "Keriting"
4     elif 14<= card_index <= 26:
5         return "Hati"
6     elif 27<= card_index <= 39:
7         return "Wajik"
8     elif 40<= card_index <= 52:
9         return "Sekop"
10    else:
11        return None

```

Source Code 25: Tipe Kartu

Fungsi diatas untuk mentukan tipe dari kartu agar komputer dapat menentukan dan mencocokan kartu dari player, penentuan tipe kartu tersebut berdasarkan index kartu atau urutan label kartu.

```

1 def comparedCard(card1, card2):
2
3     value1 = cardName.index(card1)
4     value2 = cardName.index(card2)
5
6     type1 = cardType(value1)
7     type2 = cardType(value2)
8
9     if type1 is not None and type1 == type2:
10         if value1 > value2:
11             return 1
12         elif value1 < value2:
13             return -1
14         else:
15             return 0
16     else:
17         return None
18

```

Source Code 26: Compare Card

Fungsi diatas dilakukan untuk mengkomparasi kedua kartu, yaitu kartu dari player 1 dan player 2. Hal itu dengan memastikan kartu setipe dan membandingkan nilai dari kartu berdasarkan index kartu.

```

1     def predik(bgr_frames, cardCoor):
2         storedCard = []
3
4         for i, bgr_frame in enumerate(bgr_frames):
5             X = []
6
7             img = cv2.resize(bgr_frame, (128, 128))
8             img = img.astype('float32') / 255 # Normalize to [0,
9             1]
10            X.append(img)
11
12            X = np.array(X)
13            X = X.astype('float32')
14
15            hs = model.predict(X, verbose = 0)
16            n = np.max(np.where(hs== hs.max()))
17
18
19            cv2.putText(CardRes_bg, f'{cardName[n]}{{:.2f}}'.
format(hs[0,n])}, cardCoor, cv2.FONT_HERSHEY_SIMPLEX,
0.5, font_color, 1)
20            storedCard.append(cardName[n])
21

```

```
22     return storedCard  
23
```

Source Code 27: Predict Card

Fungsi Diatas adalah untuk mendeteksi kartu yang dikeluarkan. Prediksi ini dilakukan secara real-time dan kontinyu dan mengembalikan hasil prediksi kartu terbaiknya.

```
1 def decider(storedCard1, storedCard2):  
2     global turn  
3     global P1_Cardleft  
4     global P2_Cardleft  
5     global debounce_timer  
6  
7     if len(storedCard1) == 1 and len(storedCard2) == 1:  
8         winner = comparedCard(storedCard1[0], storedCard2[0])  
9         if winner == 1:  
10             cv2.putText(frame, f"{storedCard1[0]} wins!",  
11                         (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)  
12             turn = 1  
13             if debounce_timer <= 0:  
14                 P1_Cardleft = P1_Cardleft - 1  
15                 P2_Cardleft = P2_Cardleft - 1  
16                 debounce_timer = 200  
17             else:  
18                 debounce_timer -= 1  
19  
20         elif winner == -1:  
21             cv2.putText(frame, f"{storedCard2[0]} wins!",  
22                         (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)  
23             turn = 2  
24             if debounce_timer <= 0:  
25                 P1_Cardleft = P1_Cardleft - 1  
26                 P2_Cardleft = P2_Cardleft - 1  
27                 debounce_timer = 200  
28             else:  
29                 debounce_timer -= 1  
30  
31         else:  
32             cv2.putText(stat_bg, "Card not Eligible", (10,  
33                         20), cv2.FONT_HERSHEY_SIMPLEX, 1, font_color, 1)  
34             print("Card not eligible")  
35             print(comparedCard(storedCard1[0], storedCard2  
36                         [0]))
```

Source Code 28: Decider

Fungsi diatas adalah untuk menentukan pemenang dari round tersebut berdasarkan hasil komparasi. Setelah ditentukan nilai kartu mana yang terbesar, akan dilakukan pengurangan pada jumlah kartu yang dimiliki dan menentukan giliran siapa yang dapat melangkah lebih dahulu pada round selanjutnya. Diberikan timer agar proses decrement hanya terjadi sekali karena decider ini berjalan pada loop yang bersifat kontinyu atau akan terbaca terus

```
1 font_color = (255, 255, 255) # White color in BGR format
```

```

2
3 cards_without_tutup = [card for card in cardName if card != "Kartu Tutup"]
4 randomCard = random.choice(cards_without_tutup)
5 indexRandomCard = cardName.index(randomCard)
6 typeRandomCard = cardType(indexRandomCard)
7
8 turn = 0
9 P1_Cardleft = 5
10 P2_Cardleft = 5
11 debounce_timer = 0
12
13 Menu_state = True
14 Closing_state = 0
15 status = 0
16

```

Source Code 29: Initialization

Dilakukan inisialisasi untuk menentukan beberapa hal. Ditentukan warna font berupa RGB code warna putih. Random Card yang akan digunakan sebagai Dealer Card atau kartu bukaan. Ditentukan juga posisi giliran yang dapat bergerak terlebih dahulu, sisa kartu, dan timer penahanan scan berkelanjutan. Ditentukan posisi state yang berjalan. Menu state langsung bersifat True agar langsung berjalan terlebih dahulu dan status untuk inisialisasi pemenang.

4.6.1 Main loop

```

1     if Menu_state:
2         menu.menuState()
3         Menu_state = False
4

```

Source Code 30: Menu State

Pada awal loop, langsung dijalankan menu state untuk menjalankan halaman menu dan mengubah kondisinya agar tidak kembali ke halaman menu.

```

1 mainwin = cv2.imread('asset/grass.jpg')
2 mainwin = cv2.resize(mainwin, (853,640))
3
4 # DisScore_bg = np.ones((100,193,3), dtype=np.uint8)
5 DisScore_bg = cv2.imread('asset/dirt.jpeg')
6 DisScore_bg = cv2.resize(DisScore_bg, (187,154))
7 DisScore_bg = border(DisScore_bg)
8 cv2.putText(DisScore_bg, "Card Left", (48, 25), cv2.
9 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
10 cv2.putText(DisScore_bg, "P1", (30, 69), cv2.
11 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
12 cv2.putText(DisScore_bg, "P2", (130, 69), cv2.
13 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
14 cv2.putText(DisScore_bg, f"{P1_Cardleft}", (20, 140), cv2.
15 FONT_HERSHEY_SIMPLEX, 2.5, font_color, 1)
16 cv2.putText(DisScore_bg, f"{P2_Cardleft}", (120, 140),
17 cv2.FONT_HERSHEY_SIMPLEX, 2.5, font_color, 1)
18

```

```

14     CardRes_bg = cv2.imread('asset/dirt.jpeg')
15     CardRes_bg = cv2.resize(CardRes_bg, (629,124))
16     CardRes_bg = border(CardRes_bg)
17     cv2.putText(CardRes_bg, "P1 Card", (110, 30), cv2.
18 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
19     cv2.putText(CardRes_bg, "P2 Card", (440, 30), cv2.
20 FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1)
21
22     stat_bg = cv2.imread('asset/dirt.jpeg')
23     stat_bg = cv2.resize(stat_bg, (187,284))
24     stat_bg = border(stat_bg)

```

Source Code 31: Side Window

inisialisasi diatas adalah pembuatan UI pendukung untuk bagian hasil deteksi, sisa kartu, dan status/keterangan yang sedang berjalan. Hal itu dibuat dengan membuat main window(mainwin) dengan menentukan ukurannya. UI pendukung lainnya juga ditentukan ukuran yang sesuai dengan penulisan text. Pada UI juga diberikan background dari asset.

```

1     biner = binerimg(frame)
2     kontur = frame.copy()
3     height, width, _ = frame.shape
4     midPoint = width // 2
5     cv2.line(frame, (midPoint,0), (midPoint, height),
6     (0,0,255), 5)
7
8     p1side = biner[:, :midPoint]
9     p2side = biner[:, midPoint:]

```

Source Code 32: Determining each player side

Inisialisasi diatas digunakan untuk membagi frame camera menjadi 2, yaitu sisi player 1 dan sisi player 2. Hal itu dilakukan dengan membagi frame menjadi dua dan menandainya dengan garis merah vertikal sebagai pembatas.

```

1     cropped_frames1, cardCoor1 = getContours(p1side, kontur,
2     draw=True)
3     cropped_frames2, cardCoor2 = getContours(p2side, kontur,
4     draw=True)
5
6     bgr_frames1 = [cv2.cvtColor(cropped_frame, cv2.
7     COLOR_GRAY2BGR) for cropped_frame in cropped_frames1]
8     bgr_frames2 = [cv2.cvtColor(cropped_frame, cv2.
9     COLOR_GRAY2BGR) for cropped_frame in cropped_frames2]

```

Source Code 33: Detection each side

Digunakan fungsi fungsi yang telah digunakan untuk mendeteksi kartu, yaitu mendapatkan contour, dan memprediksi kartu. Sebelum kartu diprediksi, diubah terlebih dahulu ke format BGR dikarenakan model prediksi yang digunakan diharuskan berformat BGR.

```

1    ## POSISI TURN
2    if turn == 1:
3        print("Player 1 Move First")
4        cv2.putText(stat_bg, "Player 1 Move First", (10, 20),
5        cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
6        if storedCard2 and not storedCard1:
7            cv2.putText(stat_bg, "Its not your turn", (10,
270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
8        else :
9            decider(storedCard1, storedCard2)
10       elif turn == 2:
11           cv2.putText(stat_bg, "Player 2 Move First", (10, 20),
12           cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
13           if storedCard1 and not storedCard2:
14               cv2.putText(stat_bg, "Its not your turn", (10,
270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
15           else :
16               decider(storedCard1, storedCard2)
17       elif turn == 0:
18           cv2.putText(frame, "Dealer Card", (253, 50), cv2.
FONT_HERSHEY_SIMPLEX, 1, font_color, 1, cv2.LINE_4)
19           cv2.putText(frame, f"{randomCard}", (255, 80), cv2.
FONT_HERSHEY_SIMPLEX, 0.7, font_color, 1, cv2.LINE_4)
20           if storedCard1 and storedCard2:
21               iCardP1 = cardName.index(storedCard1[0])
22               # print("value 1 =", value1)
23               iCardP2 = cardName.index(storedCard2[0])
24               # print("value 2 =", value2)
25
26               type1 = cardType(iCardP1)
27               # print("type 1= "+ type1)
28               type2 = cardType(iCardP2)
29               # print("type 2= " + type2)
30               if iCardP1 or iCardP2 == indexRandomCard:
31                   print("That card is a Dealer card")
32                   print("Please Draw Another Card")
33
34               if type1 == type2 == typeRandomCard:
35                   if iCardP1 > iCardP2:
36                       print("Player 1 Win")
37                       cv2.putText(stat_bg, "Player 1 Win", (10,
270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
38                       cv2.putText(frame, "Player 1 Win", (230,
400), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
39                       turn = 1
40                       if debounce_timer <= 0:
41                           P1_Cardleft = P1_Cardleft - 1
42                           P2_Cardleft = P2_Cardleft - 1
43                           debounce_timer = 200
44                       else:
45                           debounce_timer -= 1
46
47               elif iCardP1 < iCardP2:
48                   print("Player 2 Win")
49                   cv2.putText(stat_bg, "Player 2 Win", (10,
270), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)

```

```

48             cv2.putText(frame, "Player 2 Win", (230,
49     400), cv2.FONT_HERSHEY_SIMPLEX, 0.5, font_color, 1)
50             turn = 2
51             if debounce_timer <= 0:
52                 P1_Cardleft = P1_Cardleft - 1
53                 P2_Cardleft = P2_Cardleft - 1
54                 debounce_timer = 200
55             else:
56                 debounce_timer -= 1
57
58             else:
59                 print("Something Wrong")
60             else:
61                 print("Card not Eligible")

```

Source Code 34: Menentukan Giliran

Pada Turn 1 dan 2 adalah menentukan giliran siapa yang dapat maju terlebih dahulu dari hasil sebelumnya. Pada Turn 0, cukup berbeda dikarena terdapat kondisi tambahan yaitu kartu bukaan. Pada turn 0, kedua belah pihak dapat maju terlebih dahulu dan mencocokkan tipenya dengan kartu bukaan tersebut. Dari hasil tersebut, menentukan turn siapa yang dapat berjalan duluan.

```

1     if storedCard1:
2         if storedCard1[0]== "Kartu Tutup":
3             print("Player 1, please draw a Card")
4             cv2.putText(stat_bg, "Player 1, please draw a
Card", (10, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
font_color, 1)
5             if debounce_timer <= 0:
6                 P1_Cardleft = P1_Cardleft + 1
7                 debounce_timer = 200
8             else:
9                 debounce_timer -= 1
10            elif storedCard2:
11                if storedCard2[0]== "Kartu Tutup":
12                    print("Player 2, please draw a Card")
13                    cv2.putText(stat_bg, "Player 2, please draw a
Card", (10, 270), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
font_color, 1)
14                    if debounce_timer <= 0:
15                        P2_Cardleft = P2_Cardleft + 1
16                        debounce_timer = 200
17                    else:
18                        debounce_timer -= 1

```

Source Code 35: Draw Card

Kondisi diatas adalah jika kartu yang dideteksi berupa "Kartu tutup" yang mengisyaratkan untuk mengambil/draw kartu tambahan sesuai rule yang berjalan. Dengan draw kartu, maka dilakukan penambahan pada banyaknya kartu yang dipunya.

```
1     if P1_Cardleft == 0:
```

```

2         print("Player 1 Win!")
3         Closing_state = 1
4         status = 1
5     elif P2_Cardleft == 0:
6         print("Player 2 Win!")
7         Closing_state = 1
8         status = 2
9     elif P1_Cardleft == 0 and P2_Cardleft == 0:
10        print("Draw")
11        Closing_state = 1
12        status = 3
13

```

Source Code 36: Win Condition

Pengkonsdisikan diatas untuk menentukan Pemenang dari game. Hal itu berdasarkan jumlah kartu yang tersisa.

```

1 cv2.imshow('asli', frame)
2 mainwin[0 : frame.shape[0], 0 : frame.shape[1]] = frame
3 mainwin[10: 170, 650: 843] = DisScore_bg
4 mainwin[490: 620, 5: 640] = CardRes_bg
5 mainwin[190: 480, 650: 843] = stat_bg
6 cv2.imshow("Main",mainwin)
7

```

Source Code 37: Visual Window

Inisialisasi diatas digunakan untuk mengatur posisi window dari frame player window dan side window ke dalam mainwin.

```

1 if Closing_state == 1:
2     close.closeState(status)
3     Closing_state = 2
4
5 if (cv2.waitKey(1) == ord('q')) or (Closing_state == 2):
6     break
7

```

Source Code 38: Closing

Dari Hasil Win Condition, akan menjalankan halaman Close dan setelah menjalankan halaman close, akan menutup halaman dari game keseluruhan

5 Demo

Berikut adalah hasil dari game yang dijalankan dengan game flow, yaitu Menu State, Game State, dan Closing State. Pada Game State diasumsikan player sudah membagikan kartu dengan 5 kartu setiap playernya. Berjalannya Game kartu sesuai dengan aturan yang telah dibuat.



Figure 2: Halaman Menu



Figure 3: Halaman Credit

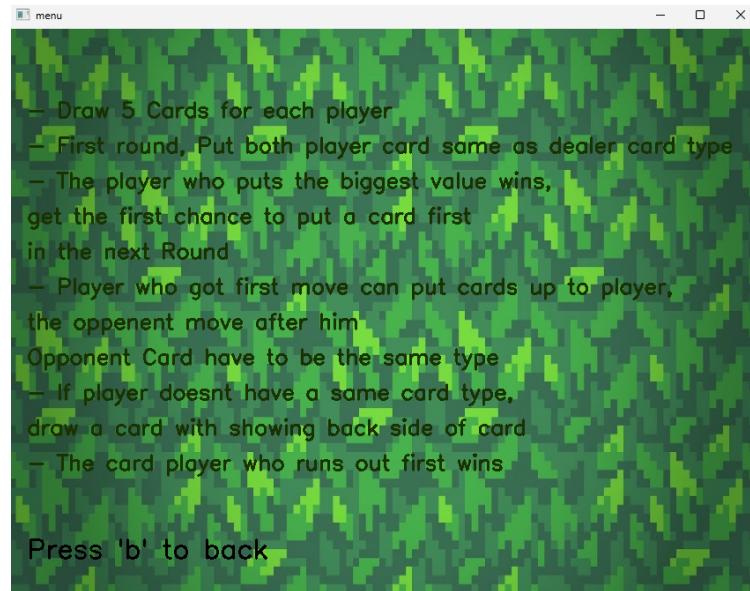


Figure 4: Halaman How to Play



Figure 5: Game Berjalan (komparasi normal)

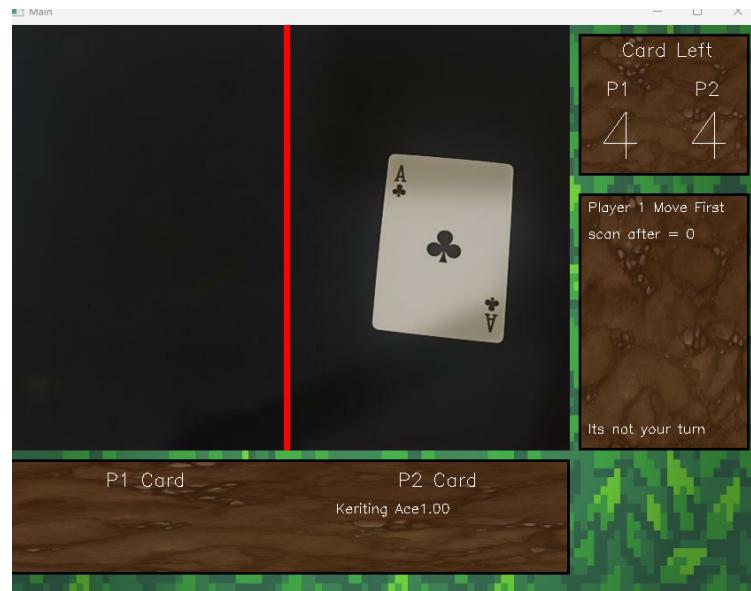


Figure 6: Aturan Turn



Figure 7: Kartu tutup untuk Draw

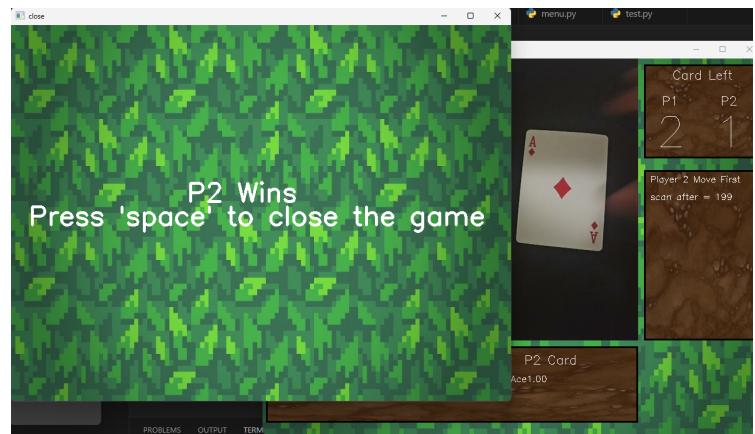


Figure 8: Win Condition

6 Penutup

6.1 Kesimpulan

Dari Pembuatan Final Project Pengolahan Citra dan Video, dapat mengimplementasikan program python dengan untuk melakukan pemrosesan citra dan mengimplementasikan machine learning didalamnya. Dengan menggunakan library OpenCV, dapat membantu pemrosesan citra, seperti mengubah warna dan format citra, mendekripsi tepi, deteksi contour, dan bentuk modifikasi lainnya. Dengan hasil pengolahan citra tersebut, dapat dipadukan dengan machine learning khususnya CNN. Dengan penggunaan CNN, komputer dapat mengetahui citra yang dideteksi sesuai dengan labelnya. Hal ini dimudahkan dengan hasil pemrosesan citra dengan OpenCV. Dengan menerapkan kedua konsep tersebut, dapat dikreasikan untuk membuat sebuah game "Minuman" menggunakan penyusunan algoritma python.

6.2 Link Source Code

GitHub: <https://github.com/Roditu/PlayingCardDetection-DrinkinGames.git>