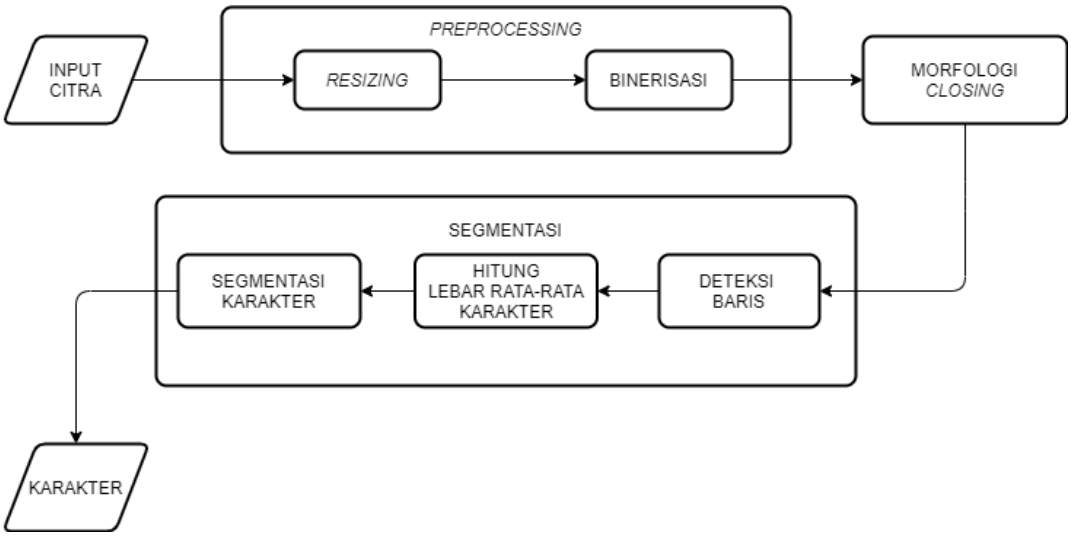
BAB 3
METODE PENELITIAN

3.1 Gambaran Umum Metode Penelitian

Pada penelitian ini dilakukan beberapa tahapan untuk membuat model segmentasi karakter pada citra tulisan tangan seperti dapat dilihat pada gambar 3.1.



Gambar 3.1 Gambaran Umum Tahapan Penelitian

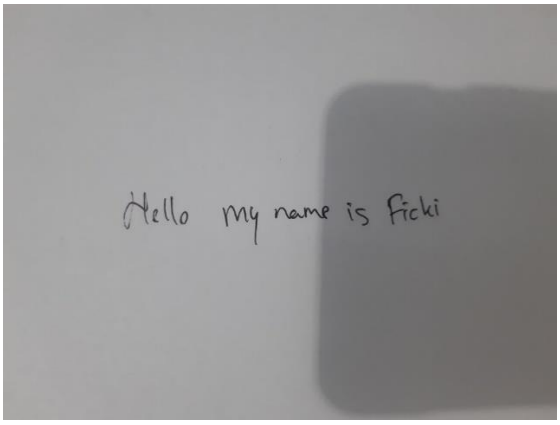
Tahapan pertama yang dilakukan adalah *preprocessing*, dimana citra di-*resize* ke ukuran 1320x(1320**H*/*W*) piksel. *H* merupakan ukuran citra input awal dan *W* merupakan lebar ukuran citra input awal. Setelah di-*resize* kemudian citra dikonversi menjadi hitam putih atau binerisasi. Citra yang sudah di-binerisasi kemudian dilakukan perataan histogram dengan metode *adaptive threshold*. Setelah melalui tahap *preprocessing*, selanjutnya citra di perlembut dengan metode morfologi *closing*, penggunaan metode ini ditujukan agar citra lebih jelas dengan menutupi lubang kecil pada kontur yang dianggap *noise* serta meleburkan garis tipis. Setelah dilakukan *morphological closing*, citra kemudian masuk ke tahapan akhir yaitu segmentasi.

Tahap segmentasi dimulai dari pendeteksian garis dengan memberi garis pada baris yang ditemukan biner 1. Setelah dilakukan deteksi garis, selanjutnya

adalah menghitung rata-rata lebar karakter. Rata-rata lebar karakter digunakan untuk deteksi kata. Setelah mendapatkan lebar karakter, proses segmentasi dilakukan dengan mendasarkan pada kontur dan peng-aplikasi-an metode *chaincode contour* sehingga akan menghasilkan segmentasi citra tulisan tangan berupa citra karakter dengan ukuran 28x28piksel.

3.2 Input Citra Tulisan Tangan

Input citra yang digunakan pada penelitian ini adalah citra tulisan tangan pribadi yang merupakan tulisan tangan penulis dan diambil menggunakan kamera *handphone* dengan hasil format jpeg dan berukuran 1152x864 piksel. Gambar 3.2 menunjukkan citra tulisan tangan yang digunakan.



Gambar 3.2 Citra tulisan tangan yang digunakan

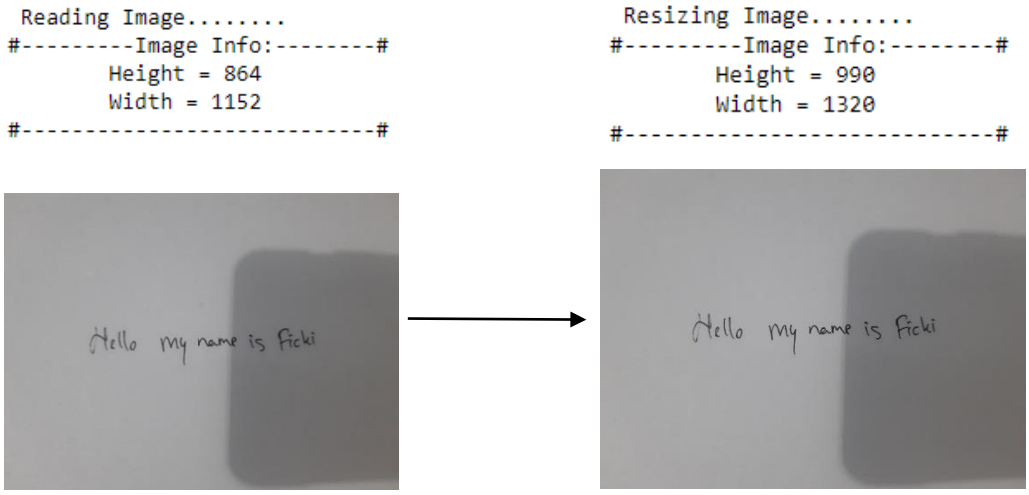
Seperti dapat dilihat pada gambar 3.2, masih terdapat ketidaksesuaian pada latar belakang akibat pencahayaan pada saat proses akuisisi data. Sehingga diperlukan tahapan selanjutnya untuk memperbaiki kualitas citra.

3.3 *Preprocessing* Citra Tulisan Tangan

Preprocessing adalah tahapan awal pengolahan data yang dilakukan terhadap citra tulisan tangan. Tahapan ini bertujuan untuk memperbaiki citra dan menekan distorsi yang tidak diinginkan atau meningkatkan fitur penting citra. Tahap ini terdiri dari 2 proses yaitu *resizing*, dan binerisasi.

3.3.1 *Resizing*

Citra yang berukuran bebas, di-*resize* untuk mempermudah proses segmentasi. Ukuran citra yang digunakan yaitu lebar 1320 dan tinggi khusus (1320 dikali tinggi citra awal dibagi lebar citra awal) dalam satuan piksel. Gambar 3.3 menunjukkan perubahan ukuran citra setelah di-*resize*.



(a) (b)

Gambar 3.3 *Resizing* citra : (a) Citra input; (b) Citra *resize*

Pada gambar 3.3, proses *resizing* dilakukan terhadap citra yang berukuran 1152x864 piksel. Ukuran citra awal kemudian diubah menjadi 1320 piksel untuk lebar citra sedangkan tingginya menjadi 990 piksel dengan menggunakan perhitungan sebagai berikut:

$$H = \frac{1320\ x\ 864}{1152} \tag{3.1}$$

Proses *resizing* menggunakan pseudocode berikut :

```
src_img = cv2.resize(copy, dsize =(1320, int(1320*height/width)),
interpolation = cv2.INTER_AREA)
```

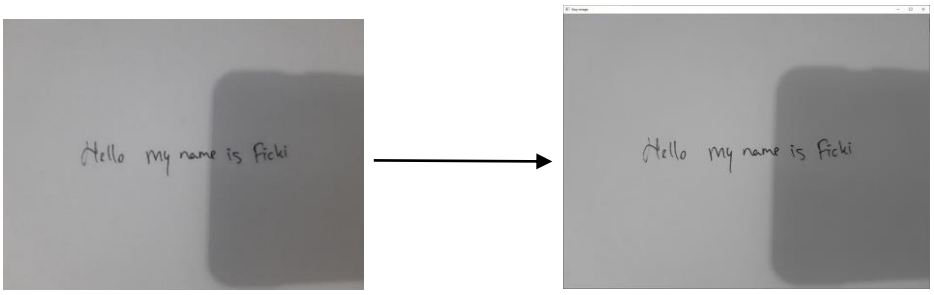
Kode program di atas menggunakan metode *resizing* yang telah disediakan oleh *library* opencv. Interpolasi dilakukan untuk membuat titik-titik baru pada piksel yang bertujuan untuk memperlebar citra.

3.3.2 Binerisasi

Citra yang sudah di-*resize*, kemudian dikonversi menjadi hitam dan putih, proses ini digunakan untuk membantu pengenalan citra sehingga citra hanya berisi 0 atau 1 pada tiap-tiap piksel. Proses binerisasi dimulai dengan konversi warna citra menggunakan metode *grayscale* milik *library* opencv dengan formula sebagai berikut:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \quad (3.2)$$

Dimana Y merupakan output dari konversi *grayscale* citra, R untuk pengubahan warna merah, G untuk pengubahan warna hijau, dan B untuk pengubahan warna biru. Gambar 3.4 menunjukkan citra yang sudah dilakukan perubahan menjadi *grayscale*.



(a) (b)

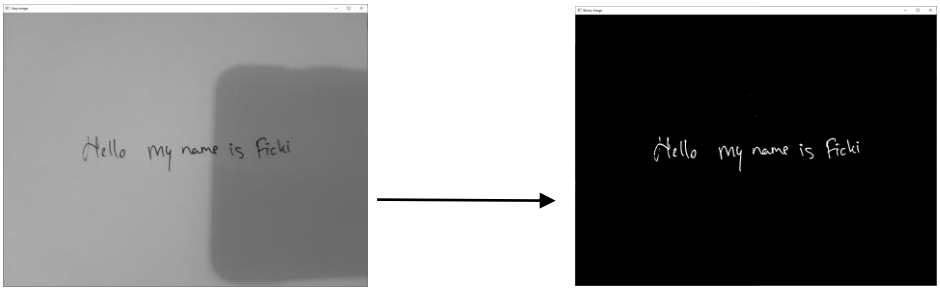
Gambar 3.4 Citra *grayscale* : (a) Citra *resize*; (b) Citra *grayscale*

Pada gambar 3.4, perubahan pada citra tidak terlalu tampak signifikan karena citra input sudah berwarna keabuan. Setelah citra menjadi *grayscale*, kemudian citra di-binerisasi dengan rumus sebagai berikut:

$$B = cv.adaptThresh(src, mxVal, adaptMethod, threshType, blockSize, C) \quad (3.3)$$

Dimana B merupakan output binerisasi, cv.adaptThresh adalah metode *library* opencv untuk binerisasi, *src* adalah sumber citra yang akan di-binerisasi, mxVal adalah nilai yang dituju untuk penemuan nilai ambang yang diatas rata-rata, adaptMethod adalah metode adaptif yang digunakan, threshType adalah tipe ambang yang digunakan, blockSize adalah ukuran piksel atau area yang digunakan

untuk mencari rata-rata ambang, C adalah konstanta yang digunakan untuk mengurangi nilai ambang rata-rata. Gambar 3.5 menunjukkan citra yang sudah di-binerisasi secara *invers*



(a) (b)

Gambar 3.5 Binerisasi citra : (a) Citra *grayscale;* (b) Citra biner

Proses binerisasi menggunakan *pseudocode* berikut :

```
bin_img =
cv2.adaptiveThreshold(grey_img,255,cv2.ADAPTIVE_THRESH_MEAN_C,c
v2.THRESH_BINARY_INV,21,20)
bin_img1 = bin_img.copy()
bin_img2 = bin_img.copy()
```

Kode program di atas merupakan proses binerisasi dengan menggunakan metode perataan histogram *adaptive threshold* karena pada kasus ini terdapat ketidaksesuaian latar karena intensitas cahaya. Metode *adaptive threshold* dipilih karena dapat mencari nilai rata-rata ambang sesuai dengan blok yang ditentukan. Setelah didapatkan nilai rata-rata ambang pada blok tersebut, nilai rata-rata dikurangi sebesar 20, kemudian nilai ambang pada tiap piksel diubah menjadi 255 ketika nilai ambangnya diatas nilai rata-rata. Kemudian binerisasi ini menggunakan tipe *thresholding inverted* yang membuat warna hitam ke putih dan sebaliknya. Citra biner ini juga digunakan untuk tahap berikutnya.

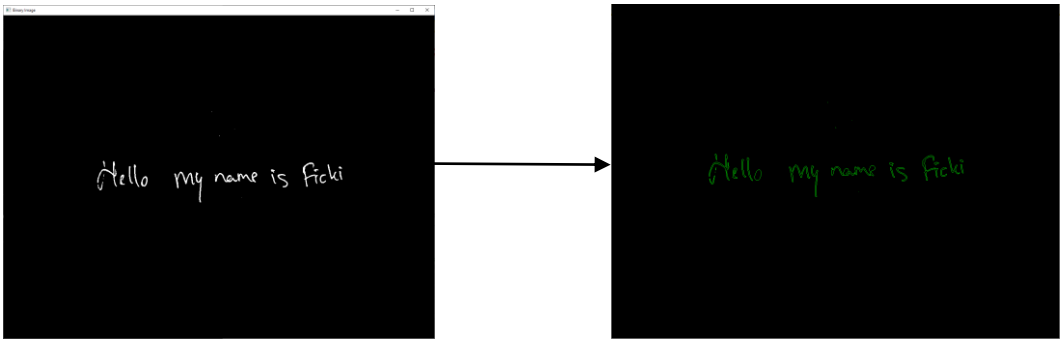
3.4 Morfologi *Closing* Citra Tulisan Tangan

Morfologi *closing* digunakan untuk menyambung bagian tulisan yang terputus dengan memperbesar batas luar dari objek *foreground* dan juga menutup lubang kecil yang terletak ditengah objek dengan rumus :

$$A \bullet S = (A \oplus S) \otimes S \tag{3.4}$$

A merupakan citra biner dan S merupakan *structuring element*. Simbol $\oplus$ menandakan erosi dan simbol $\otimes$ untuk dilatasi.

Gambar 3.6 menunjukkan citra yang diterapkan morfologi *closing.*



(a) (b)

Gambar 3.6 Morfologi *Closing* : (a) Citra biner; (b) Citra morfologi *closing*

Morfologi *closing* menggunakan *pseudocode* berikut :

```
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))
kernel1 = np.array([[1,0,1],[0,1,0],[1,0,1]], dtype = np.uint8)
final_thr = cv2.morphologyEx(bin_img, cv2.MORPH_CLOSE, kernel)
contr_retrival = final_thr.copy()
```

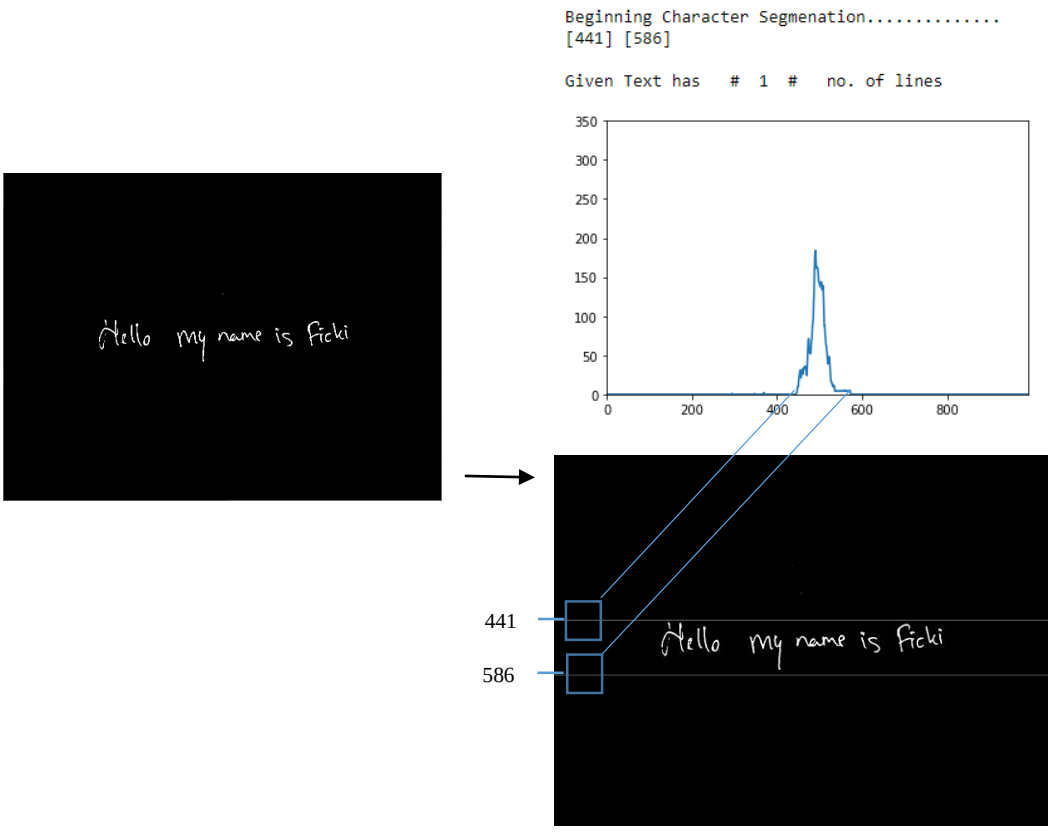
Kode program di atas menggunakan metode morfologi *closing* yang dimiliki oleh *library* opencv. Pada penelitian ini, *structuring element* pada morfologi *closing* yang digunakan peneliti berbentuk ellips untuk penyesuaian bentuk kolom citra. Setelah proses pelembutan citra dilakukan, kemudian kontur citra disimpan untuk tahapan selanjutnya.

3.5 Segmentasi Citra Tulisan Tangan

Segmentasi merupakan tahapan akhir pengolahan data citra tulisan tangan. Segmentasi pada penelitian ini akan menghasilkan citra dari tiap-tiap karakter pada citra tulisan tangan yang dikenali. Citra yang dihasilkan berupa citra dengan format jpeg dan berukuran 28x28 piksel. Tahap ini terdiri dari 3 proses yaitu deteksi baris, menghitung lebar rata-rata karakter dan segmentasi karakter.

3.5.1 Deteksi Baris

Deteksi baris dilakukan untuk membuat fokus segmentasi lebih spesifik. Deteksi baris akan memulai penggarisannya ketika menemukan nilai i (255) lebih dari 3 pada blok baris citra, penemuan ini ditandai sebagai garis atas. Kemudian garis bawah ditandai, setelah menemukan nilai i (255) berjumlah 0. Kemudian baris tersebut dicatat dan diberi garis dengan penambahan jarak 5 piksel keatas dan kebawah dari baris yang ditandai. Hasil pemberian garis atas dan garis bawah ini dihitung sebagai satu baris. Gambar 3.7 menunjukkan proses deteksi baris dari citra biner yang sudah ter-morfologi *closing*.



(a) (b)

Gambar 3.7 Deteksi Baris : (a) Citra biner morfologi *closing*; (b) Citra baris

Deteksi baris menggunakan *pseudocode* berikut:

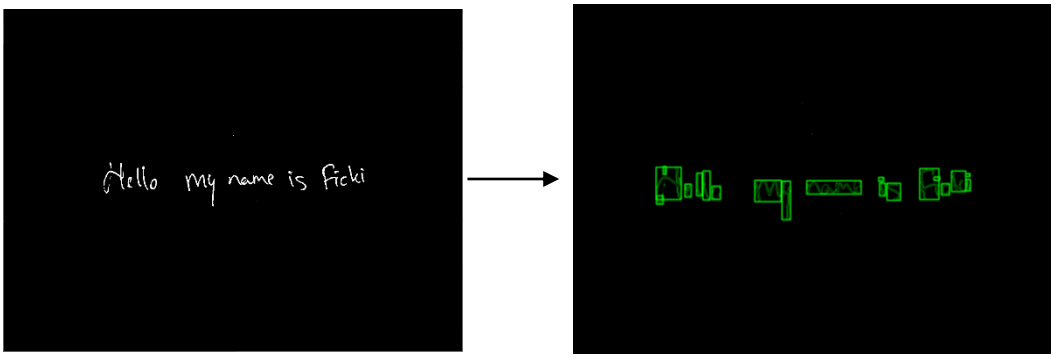
```
count_x = np.zeros(shape= (height))
for y in range(height):
      for x in range(width):
            if bin_img[y][x] == 255 :
                  count_x[y] = count_x[y]+1

upper_lines, lower_lines = line_array(count_x)
upperlines, lowerlines = refine array(upper lines, lower lines)
```

Kode program di atas akan mencari garis atas dan garis bawah pada citra biner dengan memanggil fungsi line_array, dimana fungsi ini digunakan untuk mengenali garis atas dan garis bawah dengan dipengaruhi oleh penemuan nilai x (255) sejumlah lebih dari 3 pada baris piksel hingga tidak ditemukan kembali nilai x sejumlah lebih dari 3 dalam penghitungan tersebut. Proses ini dilakukan untuk membuat proses segmentasi karakter lebih spesifik dan digunakan untuk penamaan citra karakter diakhir. Jika proses ini gagal maka program tidak dapat mengolah citra dan berhenti pada proses ini.

3.5.2 Hitung Lebar Rata-Rata Karakter

Hitung lebar rata-rata karakter digunakan untuk mendapatkan rata-rata lebar tiap objek atau karakter yang dikenali. Untuk mendapatkan lebar, teknik yang digunakan sama dengan teknik untuk segmentasi karakter. Tiap karakter yang dikenali berdasarkan penemuan nilai 0 ke 1 hingga ditemukan kembali nilai 1 ke 0. Pada teknik pencitraan kontur ini digunakan metode *chaincode*, yang mana metode ini akan membuat *boundary* dari penghubungan kontur yang dikenali. *Boundary* yang terbentuk berupa *highlight* pada bagian luar kontur serta *rectangle* selebar baris pada tiap karakter yang dikenali. Gambar 3.8 merupakan penambahan *chaincode contour* pada citra input.



(a) (b)

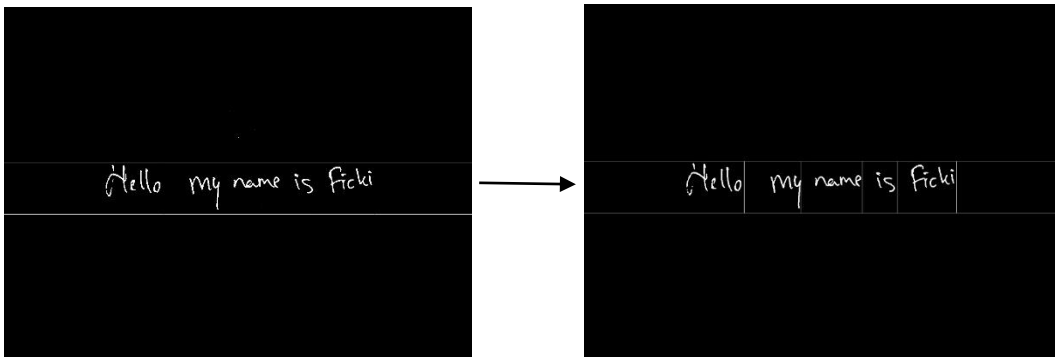
Gambar 3.8 Pengaplikasian *chaincode* pada citra : (a) Citra biner morf. *closing;* (b) Kontur citra *chaincode* dengan *rectangle box*

Proses hitung lebar rata-rata karakter menggunakan *pseudocode* berikut :

```
contours, hierarchy =
cv2.findContours(contr_retrival,cv2.RETR_EXTERNAL,cv2.CHAIN_APP
ROX_SIMPLE)
final_contr =
np.zeros((final_thr.shape[0],final_thr.shape[1],3), dtype =
np.uint8)
cv2.drawContours(src_img, contours, -1, (0,255,0), 1)

mean_lttr_width = letter_width(contours)
```

Kode program di atas menggunakan metode pencarian kontur milik *library* opencv. Setelah kontur ditemukan pada citra biner yang sudah ter-morfologi *closing*, kemudian digunakan *library* numpy untuk membentuk matriks dalam perhitungan kontur yang terhubung menggunkan *chaincode contour*, nilai rata-rata kemudian disimpan untuk mengetahui lebar rata-rata dari karakter yang dikenali dan membentuk segmentasi kata. Gambar 3.9 menunjukkan citra yang tersegmentasi per-kata.



(a) (b)

Gambar 3.9 Citra segmentasi kata : (a) Citra baris; (b) Citra baris dan kata

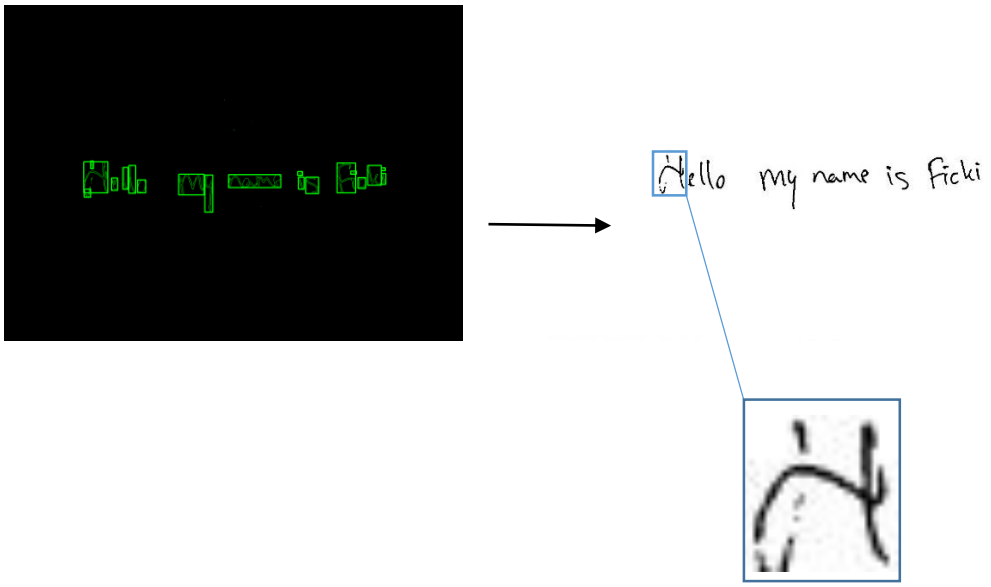
Segmentasi kata menggunakan *pseudocode* berikut :

```
x_lines = []
for i in range(len(lines_img)):
      x_lines.append(end_wrd_dtct(lines, i, bin_img,
mean_lttr_width))
for i in range(len(x_lines)):
      x_lines[i].append(width)
print(x_lines)
```

Kode program di atas merupakan proses segmentasi kata yang ditunjukkan dengan memberi garis pada setiap kata yang dikenali. Proses ini. Proses segmentasi dilakukan dengan memanggil fungsi end_wrd_dtct yang digunakan untuk mencatat titik x dari awal hingga ditemukan piksel 255 kemudian dalam jarak rata-rata lebar karakter jika tidak ditemukan piksel 255 maka garis vertikal terbentuk. Segmentasi kata ini didigunakan untuk membuat fokus segmentasi lebih spesifik dan digunakan untuk penamaan citra karakter diakhir.

3.5.3 Segmentasi Karakter

Segmentasi karakter merupakan proses akhir dari tahap segmentasi. Dimana citra yang sudah tersegmentasi baik baris dan kata, disimpan menjadi citra per tiap karakter. Segmentasi karakter ini dipotong berdasarkan lebar dari *boundary rectangle* yang sudah terbentuk pada proses sebelumnya. Citra 3.10 menunjukkan salah satu contoh citra yang sudah tersegmentasi.



(a) (b)

Gambar 3.10 Karakter Tersegmentasi : (a) Citra *closing;*
(b) Segmentasi citra karakter

Proses segmentasi menggunakan *pseudocode* berikut :

```
for i in range(len(lines)):
      letter_seg(lines_img, x_lines, i)
```

Kode program di atas menjalankan fungsi letter_seg untuk melakukan proses segmentasi karakter. Fungsi ini akan mengubah kembali citra biner *inverted* yang digunakan pada setiap tahap menjadi citra biner normal. Karakter yang tersegmentasi, berdasar *boundary rectangle* yang sudah terbentuk pada proses hitung lebar rata-rata karakter dengan menggunakan metode *chaincode contour*. Hasil dari *boundary* tersebut kemudian di-*resize* menjadi 28x28 piksel dengan penamaan citra berdasarkan baris, kata dan karakter yang dikenali.