### LAPORAN PROYEK AKHIR

### VISI KOMPUTER



Diusulkan oleh :

Hamdi Ahmadi Muzakkiy **(5112100091)**

Muhammad Nadzeri Munawar **(5112100136)**

Brillian T. Nugraha  **(5112100172)**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**SURABAYA**

**2015**

# Latar Belakang

Sistem parkir modern yang terdapat di gedung perkantoran, rumah sakit, mall, dsb. menggunakan aplikasi komputer dalam penerapannya. Selama ini, sistem parkir modern tersebut menggunakan ketik nomor plat kendaraan pada saat kendaraan masuk dan keluar dari parkiran. Tetapi ada beberapa kendala dalam penerapannya seperti kesalahan pengetikan, waktu pengetikan yang lama, serta kesalahan operator lainnya.

Oleh karena itu, perlu adanya inovasi dalam sistem parkir modern tersebut. Salah satunya yaitu menggunakan foto dalam mengenali teks plat nomor kendaraan. Tujuannya yaitu agar menghindari kesalahan pengetikan, waktu pengetikan yang lama, serta kesalahan operator lainnya.

# Rumusan Masalah

Rumusan masalah yang diangkat dalam final project ini dipaparkan sebagai berikut :

1. Bagaimana membuat aplikasi yang mampu mengambil teks dari foto plat nomor?
2. Bagaimana membuat aplikasi yang mampu menentukan warna dari foto plat nomor?

# Batasan Masalah

Permasalahan yang akan dibahas dalam final project ini memiliki beberapa batasan antara lain :

1. Algoritma yang digunakan yaitu Algoritma klasifikasi Neural Network
2. Bahasa yang digunakan yaitu Python.
3. IDE yang digunakan yaitu PyCharm.

Aplikasi yang dibuat yaitu mengambil teks dari plat nomor mobil.

# Tujuan Pembuatan Final Project

Tujuan dari pembuatan final project ini antara lain :

1. Membuat aplikasi yang mampu mendapatkan teks beserta warna dari plat nomor.

# Manfaat Final Project

Manfaat dari hasil pembuatan final project ini antara lain :

1. Menjadi solusi yang lebih baik untuk diterapkan dalam sistem parkir modern.

# Tinjauan Pustaka

Dalam final project ini, digunakan beberapa pustaka sebagai berikut :

## **Algoritma Optical Character Recognition (OCR)**

OCR adalah sebuah metode yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan. Adanya sistem pengenal huruf ini akan meningkatkan fleksibilitas ataupun kemampuan dan kecerdasan sistem komputer.

## **Bahasa Python**

Python merupakan bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sisterm operasi.

## **Tesseract**

Tesseract adalah mesin software OCR gratis yang dikembangkan di HP antara tahun 1984 dan 1994. HP dirilis ke masyarakat pada tahun 2005. Tesseract diperkenalkan pada 1995 UNLV dan saat ini dikembangkan oleh Google yang dirilis dibawah lisensi Apache. Sekarang telah dapat mengenali 6 bahasa.Pengembang dapat menggunakan Tesseract dengan font mereka sendiri dan pemetaan karakter untuk mendapatkan efisiensi yang sempurna.

## **Connected Component Analysis**

cvBlobsLib adalah perpustakaan untuk melakukan pelabelan komponen gambar bineryang terhubung. Fitur ini juga menyediakan fungsi untuk memanipulasi, filtering, dan mendapatkan hasil dari *blobs* yang telah di ekstraksikan. *Library* ini menyediakan dua fungsi dasar:

* Extract 8-*connected component* dalam biner atau grayscale gambar.
* Filter diperoleh untuk mendapatkan objek yang diinginkan di dalam gambar. Hal ini dilakukan dengan menggunakan metode Filter dari CBlobResult.

## **IDE PyCharm**

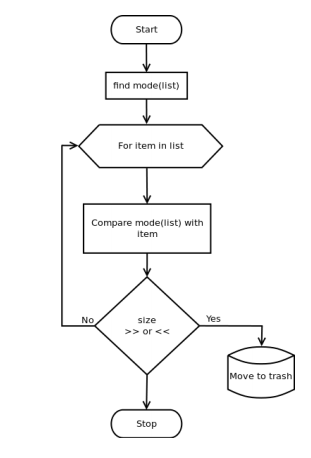
Pycharm merupakan python editor yang bisa membaca dengan baik codebase python yang secara desain adalah weak-typing. Semua keyword yang diberikan sangat akurat.

## **Segmentation**

*Image Scissoring* berfungsi sebagai pemotong gambar yang memiliki daerah dengan warna putih saja. Metode ini lebih cepat dan efisien dibandingkan metode *cropping* yang digunakan pada umumnya.

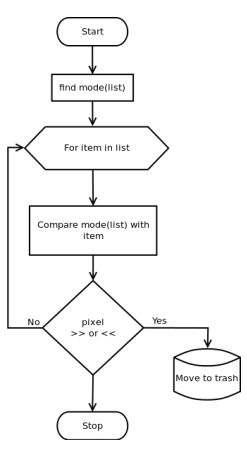
Fase *segmentation* juga melibatkan klasifikasi dari *blobs* yang dikumpulkan dan hanya mendapatkan data yang penting saja, tetapi *blobs* yang tidak diinginkan pun terkadang masih dapat melewati fase *segmentation* ini, cara menghapus *blobs* yang tidak diinginkan yaitu:

1. Eliminasi berdasarkan aspek rasio, dimana baris dan kolom akan di cek dan direkam hasilnya. Berikut adalah *flowchart*nya:



Gambar 6.1 .Diagram alir eliminasi aspek rasio

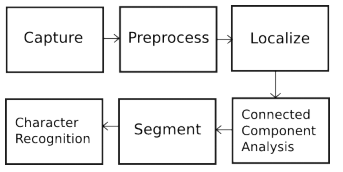
* 1. Eliminasi berdasarkan koordinasi pixelnya, dimana Algoritma ini memiliki fakta bahwa nomor lisensi ada di *plate* dalam suatu baris. Dengan efektif kita bisa mendeteksi tepi plat, dan memilih *blobs* yang berada di antara baris minimum dan maksimum koordinat. Hal ini dapat mengurangi jumlah *blobs*  yang tidak diinginkan dan membuat sistem lebih tepat.



Gambar 6.2. Diagram alir eliminasi berdasarkan koordinat piksel

# Ringkasan Isi Final Project

Aplikasi yang akan dibangun yaitu mendapatkan teks dari foto plat nomor. Cara kerjanya yaitu dari foto tampak depan kendaraan, dicari bagian plat nomornya. Setelah mendapatkan bagian plat nomor, dicari text beserta warna dari plat nomor tersebut. Berikut adalah langkah-langkah dalam mendapatkan teks dari foto plat nomor:



Gambar 7.1. Diagram alir eliminasi berdasarkan koordinat piksel

Pada gambar 1, langkah pertama yang dibuat yaitu Capture, mendapatkan gambar dari plat nomor kendaraan. Berikut adalah contoh gambar tersebut :



Gambar 7.2. Capture citra

Setelah dilakukan Capture, langkah kedua yang perlu dilakukan yaitu Preprocessing yang akan digunakan untuk meningkatkan kualitas data, sehingga klasifikasi yang akan didapatkan menjadi lebih tepat. Tahapan preprocessing yaitu ***Resize*** dan ***Convert Colour Space***, dimana ***Resize*** akan mengubah ukuran citra menjadi citra yang lebih mungkin untuk diproses (memperkecil ukuran citra), lalu:

***Convert Colour Space*** untuk mengubah citra RGB ke grayscale, di bawah ini adalah hasil gambar setelah preprocessing.



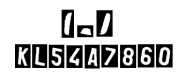
Gambar 7.3. Gray Citra

Setelah dilakukan preprocessing, langkah ketiga yang perlu dilakukan yaitu Localize. Yaitu “*binarizing the image*”, membuat pixel menjadi hitam dan putih saja. Selain itu dilakukan juga “*thresholding*” pada gambar. Tujuannya untuk memperjelas karakter dan meminimalkan kualitas gambar background, sehingga akan memudahkan untuk pengenalan karakternya yang akan digunakan untuk proses selanjutnya. Berikut adalah contoh gambar hasil dari proses Localize :



Gambar 7.4. Citra binary

Setelah dilakukan CCA, langkah kelima yang perlu dilakukan yaitu *Segmentation*. Adapun proses dari *segmentation* adalah mengambil hasil ekstraksi dari CCA dengan memisahkan masing-masing *blobs* pada gambar. Adapun algoritma yang digunakan disini yaitu menggunakan algoritma ***Image Scissoring. Image Scissoring***melakukan *segmentation* dengan cara mengamati citra secara vertikal, kemudian memotongnya pada baris dimana tidak ada pixel berwarna putih, kemudian hasil pemotongan ini akan dipindahkan ke matrix yang baru. Berikut adalah contoh hasil *segmentation*:



Gambar 7.5. Hasil segmentasi

Setelah dilakukan segmentation, langkah terakhir yang dilakukan yaitu Character Recognition menggunakan algoritma Optical Character Recognition (OCR). Hasil dari proses ini yaitu berupa teks yang dapat dibaca menggunakan ascii.

# Jadwal Pengerjaan

Timeline pengerjaan adalah sebagai berikut :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Kegiatan** | **Minggu Ke -** | | | |
| **12** | **13** | **14** | **15** |
| 1 | Pembuatan Proposal |  |  |  |  |
| 2 | Studi Literatur |  |  |  |  |
| 3 | Pembuatan Aplikasi |  |  |  |  |

Tabel 1 Time line pengerjaan

# Implementasi

Proyek Akhir pendeteksian wajah ini diimplementasikan menggunakan python. Berikut implementasi yang dilakukan.

## **Capture**

Pada sub bab ini dijelaskan implementasi gambar yang dimasukan sebelum dilakukan proses pengenalan karakter.

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5. | if 'getCarPlateNumber' in form:  fileitem = form.getvalue('location')  lenfile = len(fileitem)  with open('output\_pic.jpg', 'wb') as f:  f.write(fileitem) |

Kode Sumber 1. Implementasi Tahap Capture

## **Preprocess**

Pada sub bab ini dijelaskan implementasi *preprocess* pada gambar masukkan sebelum masuk ketahap selanjutnya. Pada tahap *preprocess* , pertama kali dilakukan adalah mengubah gambar menjadi *gray image*.

|  |  |
| --- | --- |
| 1.  2.  3.  4. | grey = prepare\_image(imgName)  size = cv.GetSize(grey)  smoothed = cv.CreateImage(size, cv.IPL\_DEPTH\_8U, 1)  cv.Smooth(grey, smoothed, cv.CV\_GAUSSIAN, 5, 5) |

Kode Sumber 2. Implementasi Gray Image

Pada Kode Sumber 2 dilakukan bluring menggunakan gaussian filter dengan filter 5x5. Setelah mendapatkan hasil *gray image*, selanjutnya mengubah citra tersebut kedalam citra *binary*. Implementasi ditunjukkan pada

|  |  |
| --- | --- |
| 1.  2.  3. | cv.CvtColor(cek,gray,cv.CV\_RGB2GRAY)  binary\_image = cv.CreateImage(cv.GetSize(gray),8,1)  cv.Threshold(gray,binary\_image,128,255,cv.CV\_THRESH\_OTSU) |

Kode Sumber 3. Implementasi Citra Binary

Setelah mendapatkan citra *binary*, masuk kedalam tahap selanjutnya yaitu *localize*.

## **Localize**

*Localize* dilakukan untuk mendapatkan area-area pada gambar. Pertama, dilakukan proses untuk mendapatkan kontur pada citra. Implementasi ditunjukkan pada Kode Sumber 4

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15. | def validate\_contour(self,c,image):  x\_min = min(pt[0] for pt in c)  x\_max = max(pt[0] for pt in c)  y\_min = min(pt[1] for pt in c)  y\_max = max(pt[1] for pt in c)  dx = x\_max - x\_min + 1  dy = y\_max - y\_min + 1  d = float(dx)/float(dy)  a = dx\*dy  i = float(len(c))/a  g = 0.3  if (len(c)>8 and a>20 and dy>8 and dy<50 and dx<50 and d>0.125 and d<8 and i>0.02):  return Box(image, x\_min, y\_min, x\_max, y\_max)  else:  return None |

Kode Sumber 4. Implementasi Kontur

Setelah mendapatkan kontur dari citra, dilakukan klastering untuk didapatkan grup area gambar. Implementasi ditunjukkan pada Kode Sumber 5

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16.  17.  18.  19.  20.  21.  22.  23 | def find\_cluster(self,bboxes):  clusters = []  for i in bboxes:  line = None  cluster = set((i,))  nb = [(b, i.dist\_to(b)) for b in bboxes if b != i and i.similar\_to(b)]  nb.sort(cmp=lambda x,y: cmp(x[1],y[1]))  for b,d in nb:  dists = [j.dist\_to(b) for j in cluster]  if min(dists) < math.sqrt(b.w\*b.w+b.h\*b.h)\*1.6:  if line is None or self.fit\_line(line, i, b):  cluster.add(b)  d = i.dist\_to(b)  if line is None and d != 0.0:  line = ((b.cx-i.cx)/d, (b.cy-i.cy)/d)  if len(cluster)>3:  clusters.append(cluster)  clusters.sort(cmp=lambda x,y: cmp(len(x),len(y)))  if len(clusters):  return [clusters[-1]] + self.find\_cluster(bboxes.difference(clusters[-1]))  else:  return [] |

Kode Sumber 5. Implementasi Klastering

Dari hasil ini, dilakukan proses segmentasi untuk mendapatkan objek plat nomor.

## **Segmentation**

Dari hasil klastering, tahap selanjutnya adalah segmentasi. Segmentasi dilakukan untuk menemukan area yang mempunyai teks. Implementasi ditunjukkan pada Kode Sumber 6

|  |  |
| --- | --- |
| 1.  2.  3.  4.  5.  6.  7.  8.  9.  10.  11.  12.  13.  14.  15.  16. | for cluster in clusters:  cv.Rectangle(result,  (int(min([c.x1 for c in cluster])),  int(min([c.y1 for c in cluster]))),  (int(max([c.x2 for c in cluster])),  int(max([c.y2 for c in cluster]))),  (0,0,255))  for idx,bbox in enumerate(cluster):  cv.Rectangle(result,  (int(bbox.x1),int(bbox.y1)), (int(bbox.x2), int(bbox.y2)), (255,0,0))  if x1 > bbox.x1:  x1 = bbox.x1  if x2 < bbox.x2:  x2 = bbox.x2  if y1 > bbox.y1:  y1 = bbox.y1  if y2 < bbox.y2:  y2 = bbox.y2  im = cek[y1:y2, x1:x2]  im=np.array(im)  self.imagePreprocessed = 'baru.jpg'  cv.imwrite(self.imagePreprocessed,im) |

Kode Sumber 6. Implementasi Segmentasi

Hasil dari proses ini adalah citra plat nomor yang berhasil ditemukan selama proses *preprocess*.

## **Character Recognition**

Pada tahap ini dilakukan pengenalan teks dari citra yang sudah di proses pada tahap preprocess. Implementasi dari *Character Ricognition* bisa dilihat pada

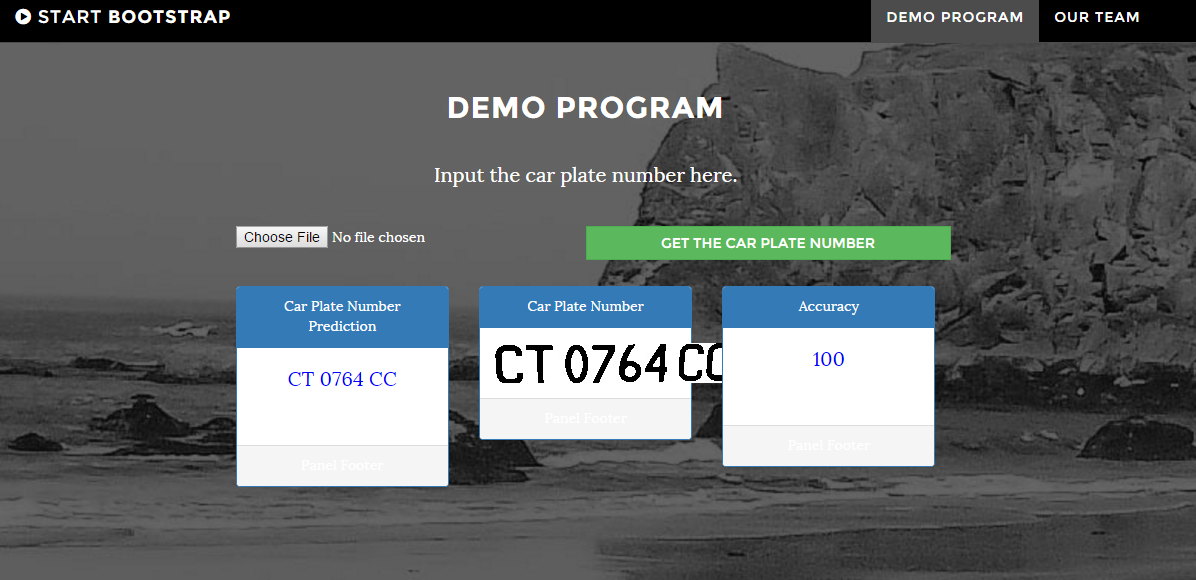
|  |  |
| --- | --- |
| 1.  2.  3.  4. | def classify(self):  openImage = Image.open(self.imagePreprocessed)  text = t.image\_to\_string(openImage)  return text |

Kode Sumber 7. Implementasi Character Ricognition

Hasil dari proses ini adalah teks berupa plat nomor yang sudah dikenali.

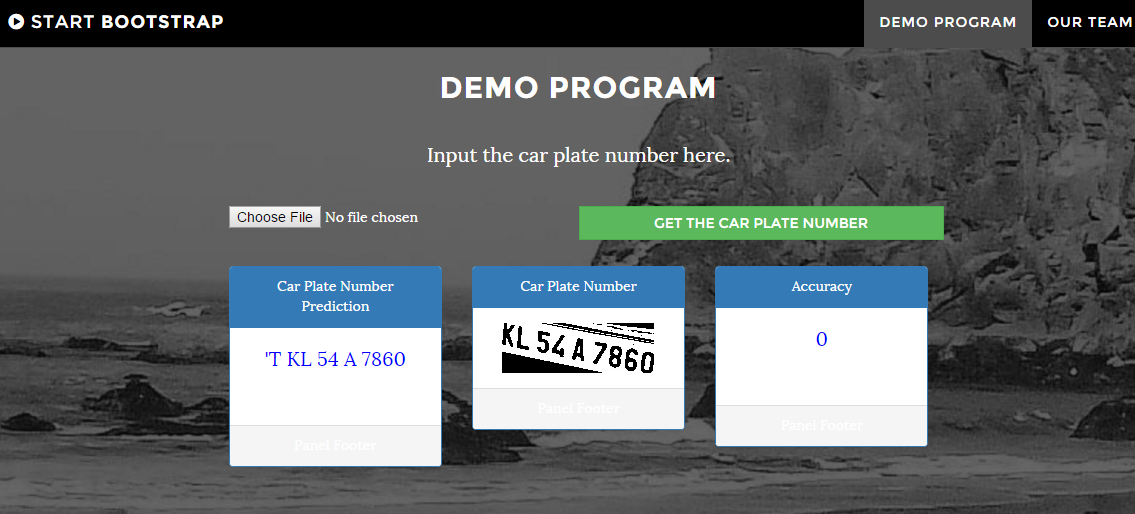
## **Result**

Hasil pengenalan karakter teks dari citra yang telah diproses menghasilkan hasil sebagai berikut.



Gambar . Hasil Character Recognition.

Pada gambar 9.1, hasil akurasi mencapai 100% dikarenakan label dari plat nomor yang telah dimasukkan sama dengan prediksi plat nomor kendaraan yang diproses oleh sistem.



Gambar . Hasil akurasi pengenalan plat nomor yang tidak pas.

Pada gambar 9.1, hasil akurasi mencapai 0% dikarenakan hasil prediksi plat nomor tercampur dengan prediksi karakter lain, sehingga hasil proses penghitungan akurasi menjadi tidak akurat ketika dimasukkan ke dalam sistem.

## **Kesimpulan dan Saran**

Berdasarkan implementasi dan hasil dari *character recognition*, didapatkan kesimpulan:

* 1. Aplikasi plate recognition berhasil dibuat, dengan akurasi yang cukup baik (70%-100%). Pada citra dengan akurasi 0%, didapatkan karena hasil prediksi gambar terjadi pencampuran dengan karakter lain n-karakter pertama sehingga mengakibatkan akurasi menjadi tidak optimal.
  2. Gambar citra mobil dan plat nomornya dapat dipisahkan dengan menggunakan deteksi tepi dan kontur.
  3. Hasil dari deteksi tepi dan kontur digunakan sebagai input untuk *clustering* yang akan memisahkan karakter-karakter yang ada didalam citra.

Saran :

* 1. Ditambahkan pengenalan warna plat nomor mobil dari citra yang di *upload* ke dalam sistem aplikasi web.
  2. Perhitungan akurasi harus bisa memperhatikan dan membuang karakter tambahan/campuran di n-karakter pertama, sehingga proses perhitungan akurasi menjadi lebih optimal.