BAB 3
METODE PENELITIAN

3.1 ALAT YANG DIGUNAKAN

3.1.1 Laptop

Perangkat keras yang digunakan dalam pengklasifikasian kendaraan mobil dan becak yang menggunakan metode *edge detection* dan *SVM* ini menggunakan laptop yang memiliki spesifikasi sebagai berikut :

Windows Edition : *Windows* 10

Processor : Intel Core i3-6100U up to 2.30 GHz

InstalledMemory (RAM): 4GB DDR4 SDRAM 2133MHz

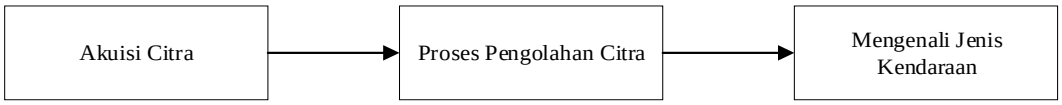
SystemType : 64-bit

3.1.2 Software

Perangkat lunak yang digunakan dalam pengklasifikasian kendaraan mobil dan becak ini menggunakan software PyCharm dan *Library OpenCV*. *PyCharm* merupakan sebuah *software* yang fungsinya digunakan dalam pemrograman komputer, yang dikhususkan untuk bahasa pemrograman *python*. *Library OpenCV* adalah sebuah *library open source* yang dikembangkan oleh *intel* yang berfokus untuk menyederhanakan sebuah programing terkait citra digital. Di dalam *OpenCV* memiliki banyak fitur, diantaranya adalah pengenalan wajah, pelacakan wajah, deteksi wajah, dan berbagai jenis metode *AI (Artificial Intellegence),* yang menyediakan berbagai macam algoritma yang sederhana terkait dengan *Computer Vision* untuk low level API.

3.2 DIAGRAM BLOK SISTEM

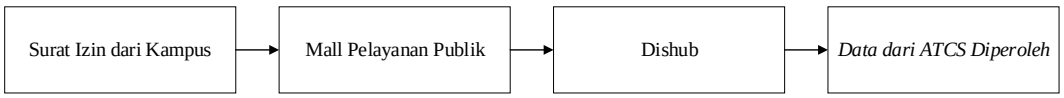
Pada gambar 3.1 adalah diagram blok yang akan dilakukan untuk mendapatkan suatu sistem yang akan dirancang



Gambar 3.1 Diagram Blok Sistem

Pada sistem ini langkah pertama adalah melakukan akuisi citra video yang didapat dari *ATCS* Purwokerto untuk mengambil citra berupa kendaraan yang melintasi area Kebon Dalem.

Proses Pengambilan video CCTV dari ATCS Banyumas dijelaskan pada Gambar 3.2



Gambar 3.2 Proses Pengambilan Data

Untuk Proses pengambilan data, data diambil dari Dinas Perhubungan Kabupaten Banyumas, langkah pertama penulis dalam pengambilan data adalah datang ke mall publik untuk bertanya syarat-syarat untuk membuat surat izin agar bisa mengambil data dari dinas perhubungan. Syaratnya adalah, fotocopy KTP satu lembar, Surat Izin dari kampus, dan foto bewarna 1 lembar, dan *hardcopy* file Skripsi. Lalu langkah selanjutnya adalah pembuatan surat izin dari kampus untuk ditujukan kepada Kepala Kesbangpol Banyumas, surat dibuat pada tanggal 27 Januari 2020. Setelah itu datang ke Mall publik untuk menyerahkan berkas persyaratan untuk pengambilan data, proses ini memakan waktu 2 minggu sampai sebulan. Setelah surat izin dari mall public didapatkan, kita dapat menyerahkannya ke Dinas Perhubungan dan menunggu untuk dikonfirmasi lagi, proses ini bisa sampai sebulan, surat baru diproses pada tanggal 20 Maret 2020 oleh bagian Keselamatan Dinas Perhubungan Kabupaten Banyumas. Dikarenakan terjadi Covid 19 maka file dari *ATCS* Banyumas dikirimkan ke email penulis pada tanggal 31 Maret 2020. Penulis mendapatkan 3 file video dari *CCTV ATCS* Kebon Dalem yang Panjang videonya berisi kurang lebih satu menit.

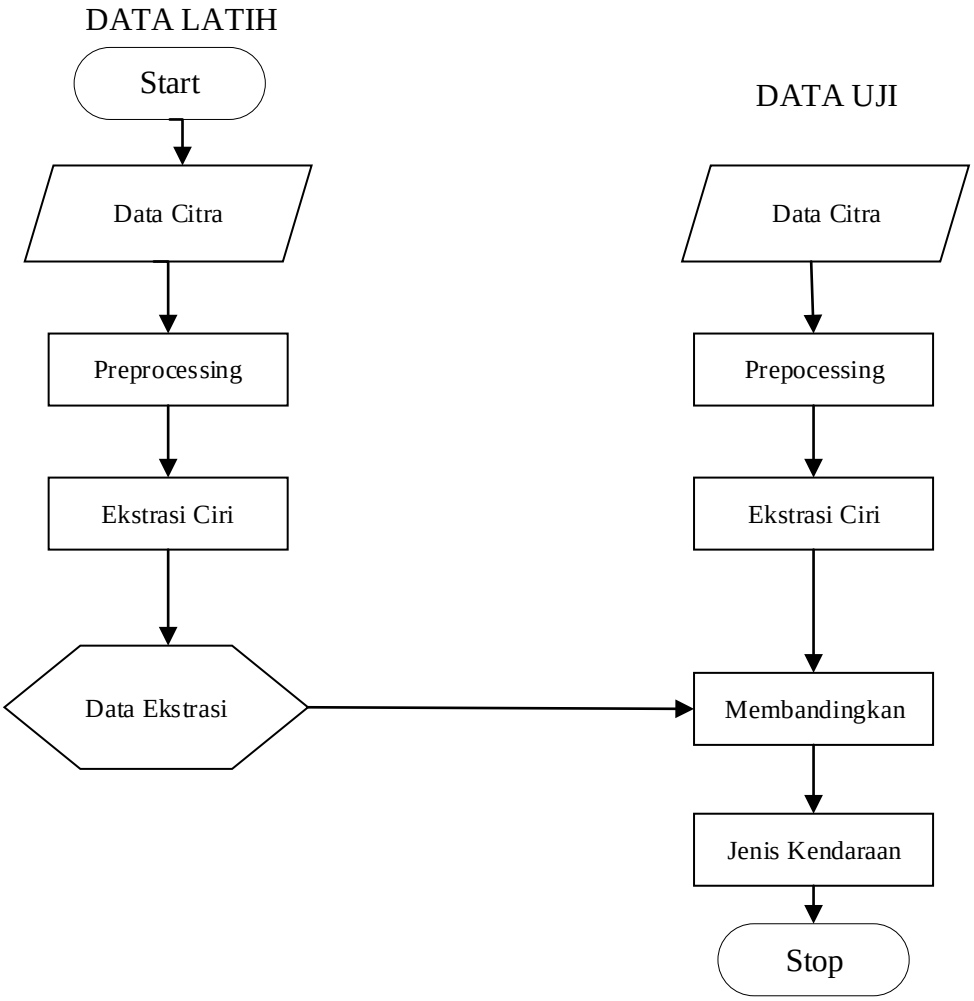
Lalu citra langsung melakukan proses pengolahan citra dengan metode *HOG* dan *SVM* untuk mendapapatkan sebuah citra yang dapat mengenali

jenis kendaraan berupa mobil atau becak. Becak sebagai *citra False* dan mobil sebagai c*itra True.*

3.3 PEMODELAN SISTEM

Dalam tahap ini dirancang Dalam tahap ini dirancang kebutuhan dalam pengklasifikasian mobil dan becak berdasarkan metode *SVM* .

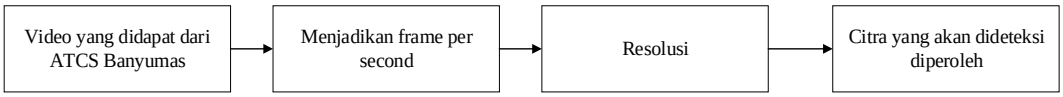
Pemodelan Sistem dalam Penelitian ini adalah :



Gambar 3.3 Pemodelan Sistem

3.3.1 Data Citra

Citra mobil pada penelitian ini, di*capture* dengan menggunakan kamera *CCTV ATCS* Kabupaten Banyumas. Pengambilan citra dilakukan di Kebon Dalem. Pengambilan gambar dilakukan di saat cuaca cerah, dan citra yang dihasilkan yaitu citra berwarna dalam format *.jpg.



Gambar 3.4 Gambar yang diperoleh

Setelah video dari *ATCS* Kabupaten Banyumas didapatkan maka langkah selanjutnya adalah menjadikan video tersebut menjadi sebuah citra, dengan cara memotong video tersebut. Pada penelitian ini didapatkan video dari ATCS, pada video pertama durasinya adalah 57 detik, frame persecondnya adalah 25, dan resolusinya 586X480 *pixel*. Pada video kedua durasinya adalah 2.44 menit, frame persecondnya adalah 25, dan resolusinya 586X480 *pixel*. Pada video ketiga durasinya 50 detik, frame persecondnya adalah 25 dan resolusinya adalah 586X480 *pixel*. Jadi pada video pertama terdapat 1430 citra, video kedua terdapat 4104 citra, dan pada video ketiga 1268 citra. Pemilihan citra diambil secara acak.

Untuk coding pemotongan video menjadi sebuah frame dapat dilihat pada gambar 3.5

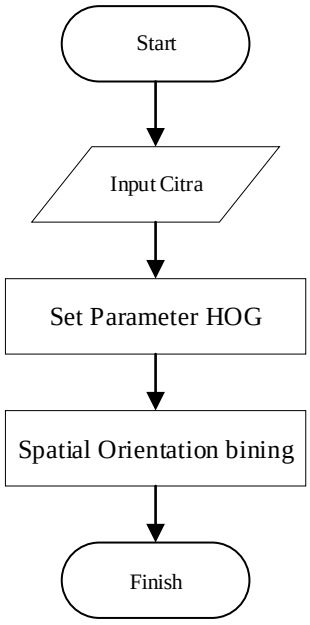
```python
import cv2
import numpy as np

vidcap = cv2.VideoCapture('kebondalem3.mp4')
success, image = vidcap.read()
src_path_pattern = "output_images_3/frame"
count = 0
success = True
while success:
 src_path = src_path_pattern+str(count)+".jpg"
 print(src_path)
 print(success)
 w = np.size(image,0)
 h = np.size(image,1)
 print(w)
 print(h)
 cv2.imwrite(src_path, image)  # save frame as JPEG file
 count += 1
 success, image = vidcap.read()
 print("Export's Completed!")

```

Gambar 3.5 Coding Pemotongan video

3.3.2 **Proses HOG**



Gambar 3. 6 Proses HOG

Proses awal dari sistem ini adalah pengolahan data akuisisi citra, pengambilan data dengan menggunakan media kamera dari *ATCS* Purwokerto. Citra tersebut dijadikan sebagai data latih atau data input yang berupa gambar bentuk mobil tampak depan dengan format *.jpg.

gambar ini adalah adalah hasil pemotongan dari video dari *ATCS* mobil dan becak. Setelah citra terbaca pada *opencv*, proses selanjutnya adalah mengubah file *feature_image* ke format RGB. Lalu proses seanjutnya adalah mengeset parameter untuk *edge detection hog* dan mengambil *hog_feature* dari *sample* sebelumnya. Proses selanjutnya adalah *spatial orientation binning*, dalam membuat histogram membutuhkan nilai dari gradien dan nilai itu akan didapat dari nilai setiap piksel didalam sebuah citra. Citra tersebut akan dibagi menjadi *cells* dengan ukuran *cell* yang telah ditentukan. Pada tiap *cell* yang ada pada citra akan dibuat sebuah histogram yang fungsinya untuk mengetahui nilai-nilai yang ada pada *cell* tersebut, dikarenakan pada setiap *cell* itu memiliki nilai yang berbeda-beda. Fungsi dari bin dalam pembuatan histogram adalah untuk mengetahui nilai-nilai dari setiap gradiennya. Dalam penelitian ini menggunakan 32 bin orientation.

```
# Set Hyperparameters
color_space = 'HLS' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 9 # HOG orientations
pix_per_cell = 8 # HOG pixels per cell
cell_per_block = 2 # HOG cells per block
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
spatial_size = (8, 8) # Spatial binning dimensions
hist_bins = 8 # Number of histogram bins
spatial_feat = True # Spatial features on or off
hist_feat = True # Histogram features on or off
hog_feat = True # HOG features on or off
#y_start_stop = [400, 650] # Min and max in y to search in
slide_window() ori
y_start_stop = [100, 500] # Min and max in y to search in
slide_window()

# Define a function to return HOG features and visualization
def get_hog_features(img, orient, pix_per_cell, cell_per_block,
vis=False, feature_vec=True):
 # Call with two outputs if vis==True
 if vis == True:
  features, hog_image = hog(img, orientations=orient,
  pixels_per_cell=(pix_per_cell, pix_per_cell),
  cells_per_block=(cell_per_block, cell_per_block),
  transform_sqrt=True,
  visualize=vis, feature_vector=feature_vec)
  return features, hog_image
# Otherwise call with one output
 else:
  features = hog(img, orientations=orient,
  pixels_per_cell=(pix_per_cell, pix_per_cell),
  cells_per_block=(cell_per_block, cell_per_block),
  transform_sqrt=True,
```

```
  visualize=vis, feature_vector=feature_vec)
  return features
```

3.3.3 Proses Ekstraksi ciri

Proses ekstraksi ciri dilakukan setelah proses preprocessing. Ekstraksi ciri merupakan langkah awal dalam melakukan klasifikasi dan interpretasi citra. Ekstraksi ciri adalah metode pengambilan ciri yang didasarkan pada karakteristik histogram citra. Histogram menunjukkan probabilitas kemunculan nilai derajat keabuan piksel pada suatu citra [10]. Proses ini berkaitan dengan klasifikasi jenis mobil dan becak ke dalam sekelompok nilai ciri yang sesuai. Proses ektraksi ciri ini menggunakan metode *hog*. Jika color space yang digunakan bukan RGB maka ada *color space* yang lain yang bisa terdeteksi, terdapat 5 kemungkinan *color space*, yaitu *HSV, LUV, HLS, YUV dan YCrCB*

```
#EXTRACT FEATURES
# Define a function to extract features from a single image window
def single_img_features(img, color_space='RGB', spatial_size=(32,
32),
                        hist_bins=32, orient=9,
                        pix_per_cell=8, cell_per_block=2,
hog_channel=0,
                        spatial_feat=True, hist_feat=True,
hog_feat=True, hog_features=None):
    #1) Define an empty list to receive features
    img_features = []
    #2) Apply color conversion if other than 'RGB'
    if color_space != 'RGB':
        if color_space == 'HSV':
            feature_image = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
        elif color_space == 'LUV':
            feature_image = cv2.cvtColor(img, cv2.COLOR_RGB2LUV)
        elif color_space == 'HLS':
            feature_image = cv2.cvtColor(img, cv2.COLOR_RGB2HLS)
        elif color_space == 'YUV':
            feature_image = cv2.cvtColor(img, cv2.COLOR_RGB2YUV)
        elif color_space == 'YCrCb':
            feature_image = cv2.cvtColor(img, cv2.COLOR_RGB2YCrCb)
    else: feature_image = np.copy(img)
    #3) Compute spatial features if flag is set
    if spatial_feat == True:
        spatial_features = bin_spatial(feature_image,
size=spatial_size)
        #4) Append features to list
        img_features.append(spatial_features)
    #5) Compute histogram features if flag is set
    if hist_feat == True:
```

```
        hist_features = color_hist(feature_image, nbins=hist_bins)
        #6) Append features to list
        img_features.append(hist_features)
    #7) Compute HOG features if flag is set
    if hog_feat == True:
        if True: #hog_features is None:
            if hog_channel == 'ALL':
                hog_features = []
                for channel in range(feature_image.shape[2]):

hog_features.extend(get_hog_features(feature_image[:,:,channel],
                                     orient, pix_per_cell,
cell_per_block,
                                     vis=False,
feature_vec=True))
            else:
                hog_features =
get_hog_features(feature_image[:,:,hog_channel], orient,
                         pix_per_cell, cell_per_block,
vis=False, feature_vec=True)
        #8) Append features to list
        img_features.append(hog_features)
```

3.3.4 Sistem Klasifikasi

Proses klasifikasi dilakukan setelah didapatkan hasil dari proses ekstrasi ciri. Hasil dari ekstrasi berupa vector ciri yang nantinya akan menjadi input pada proses klasifikasi. Klasifikasi disini menggunakan *Classify SVM*. Untuk citra positif yang terdeteksi mobil atau becak bernilai 1 dan citra negative yang tidak terdeteksi becak atau mobil bernilai -1

```
############ TRAIN CAR CLASSIFIER
# car_folders = ['train/vehicles/GTI_Far',
'train/vehicles/GTI_Right']
# car_folders = ['train/vehicles/GTI_Far']
X_train = None
X_test = None
y_train = None
y_test = None
svc = None
t = None
car_folders = None
f = None
axarr = None
image = None
heatmap = None
labels_list = None
car_folders
# notcar_folders = None
car_folders = ['train/vehicles/GTI_Far']
# car_folders = ['train/vehicles/Rickshaw']
# notcar_folders = ['train/non-vehicles/GTI']
scaled_X, y, X_scaler = prepare_images_for_processing(car_folders,
```

```
notcar_folders, "png",secondTrain=True)
# Split up data into randomized training and test sets
rand_state = np.random.randint(0, 100)
X_train, X_test, y_train, y_test = train_test_split(
    scaled_X, y, test_size=0.8, random_state=rand_state)
```

3.3.5 Analisis Bentuk

Pada proses deteksi mobil dan becak ini menggunakan parameter yang terbaik setelah melalui proses pengujian. Untuk proses deteksi ini menggunakan *sliding window* untuk mengecek per*pixel* dalam ukuran window yang berukuran 32x32 pixel.

```
############### LABELING VEHICLES
f, axarr = plt.subplots(3, 1, figsize=(60, 30))
# startIndex = random.randint(1, 1250) #ori
# startIndex = random.randint(695, 784)
# for i in range(0,3):
index = 765
print('read output_images/frame' + str(index) + '.jpg')
img = cv2.imread('output_images/frame' + str(index) + '.jpg')
img_with_label = label_vehicles(img, X_scaler)
axarr[0].imshow(img_with_label)
index = 766
print('read output_images/frame' + str(index) + '.jpg')
img = cv2.imread('output_images/frame' + str(index) + '.jpg')
img_with_label = label_vehicles(img, X_scaler)
axarr[1].imshow(img_with_label)
index = 767
print('read output_images/frame' + str(index) + '.jpg')
img = cv2.imread('output_images/frame' + str(index) + '.jpg')
img_with_label = label_vehicles(img, X_scaler)
axarr[2].imshow(img_with_label)
plt.setp([a.get_xticklabels() for a in axarr[:]], visible=False)
plt.setp([a.get_yticklabels() for a in axarr[:]], visible=False)
f.subplots_adjust(hspace=0)
plt.title("LabelVehicles");
plt.show()
```

3.2 Presentase Keberhasilan

3.4.1 Akurasi, presisi, dan *recall*

Dalam membangun sebuah sistem pendeteksian mobil dan becak ini diharapkan memiliki nilai akurasi, presisi dan recall yang baik. Untuk mengetahui seberapa baik sistem kita dilakukanlah pengukuran tingkat akurasi, presisi, dan recall dengan menggunakan rumus sebagai berikut [11]:

$$Recall\ mobil = \frac{TP}{TP+FN} \quad (3.1)$$

$$Precission\ mobil = \frac{TP}{TP+FP} \quad 3.2$$

$$Recall\ becak \quad = \frac{TN}{TN+FP} \quad 3.4$$

$$Precission\ becak = \frac{TN}{TN+FN} \quad 3.5$$

$$Accuracy \ = \frac{TP+TN}{TP+TN+FP+FN} \quad 3.6$$