## [Rubric](https://review.udacity.com/#!/rubrics/571/view) Points

## Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

### Camera Calibration

私はまず９ｘ６点のコーナを持つチェッカーボード上を**Z=0**平面、描かれた各コーナを**X**、**Y**座標とする実座標系を定義しました。

そして画面上に表示座標系を設定し、撮影した先のチェッカーボードの各コーナ９ｘ６点のピクセルを表示座標として記録しました。

次に向きや位置を異なる２０通りパターンで移動させて、各パターンにおけるコーナ９ｘ６点の表示位置を記録しました。

各パターン、各コーナの**XYZ**値はリスト**objpoints**に格納し、表示ピクセル値はリスト**imgpoints**に格納しました。

私は**objpoints**と**imgpoints**を入力とするcv2.calibrateCamera()関数を使用してカメラの較正および歪み係数を計算しました。この歪み補正をcv2.undistort()関数を使ってテスト画像に適用し、右図の結果を得ました：

I first defined a real coordinate system with a Z = 0 plane on the checkerboard with a 9x6 corner and X, Y coordinates for each drawn corner.

Then, we set the display coordinate system on the screen and recorded the pixels of 9 x 6 points on each corner of the checkerboard that we took the picture as display coordinates.

Next, the direction and position were moved in 20 different patterns, and the display position of the corner 9 x 6 points in each pattern was recorded.

Each pattern, the XYZ values of each corner are stored in the list objpoints, and the display pixel values are stored in the list imgpoints.

I calculated the camera calibration and distortion factor using the cv2.calibrateCamera () function with objpoints and imgpoints as input. This distortion correction was applied to the test image using the cv2.undistort () function, and the result of the right figure was obtained:

### Pipeline (single images)

#### 1. Provide an example of a distortion-corrected image.



#### 2. Describe how (and identify where in your code) you used color transforms, gradients or other methods to create a thresholded binary image. Provide an example of a binary image result.

後述します。

I will describe it later

#### 3. Describe how (and identify where in your code) you performed a perspective transform and provide an example of a transformed image.

私はまずundistort()関数で直進中の画像を補正して基準画像を作成した。

次に基準画像から鳥瞰画像への変換キーとなる台形エリアのコーナ４点を検討した。

それらが示すエリアが、鳥瞰図上の横位置1/4〜3/4エリアに表示されるような変換基準点の抽出を試みた。

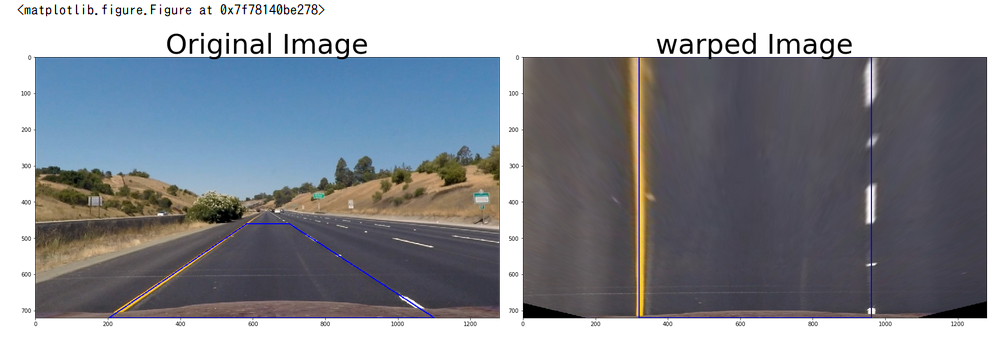
私が設定した4点の基準ポイントの組み合わせを右下表に示す。

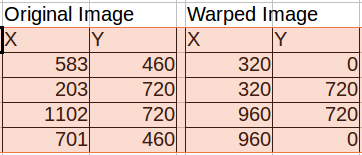
I first made a reference image by correcting the straight forward image with the undistort () function.

Next, four corners of the trapezoidal area, which is a conversion key from the reference image to the bird's-eye view image, were examined.

We attempted to extract conversion reference points such that the area indicated by them is displayed in the 1/4 to 3/4 horizontal position on the bird's eye view

The combination of the four reference points I set are shown in the lower right table.





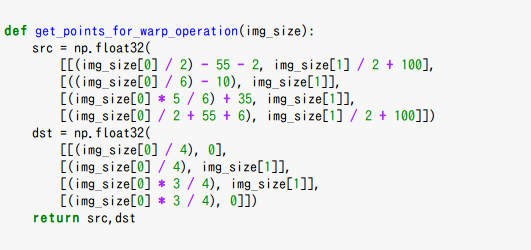
また一連の作業について得た4つの対応点を、いつでもデータを再利用可能できるように下記関数(get\_points\_for\_warp\_operation)を定義した。

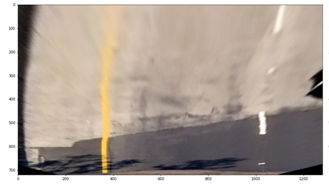
その後４つの対応点より鳥瞰画像変換のためのM行列をcv2.getPerspectiveTransform関数を用いて取得した。  
また、得たM行列と cv2.warpPerspective関数を用いて鳥瞰画像を作成できた。

In addition, the following function (get\_points\_for\_warp\_operation) is defined so that the data correspondence can be reused at any time for the four corresponding points obtained in the series of work.

Then, an M matrix for bird's-eye view image transformation was acquired from the four corresponding points using cv2.getPerspectiveTransform function.

In addition, a bird's-eye image can be created using the obtained M matrix and the cv2.warpPerspective function.





今度は、鳥瞰図より、画像の中に含まれるレーンの抽出を行います。

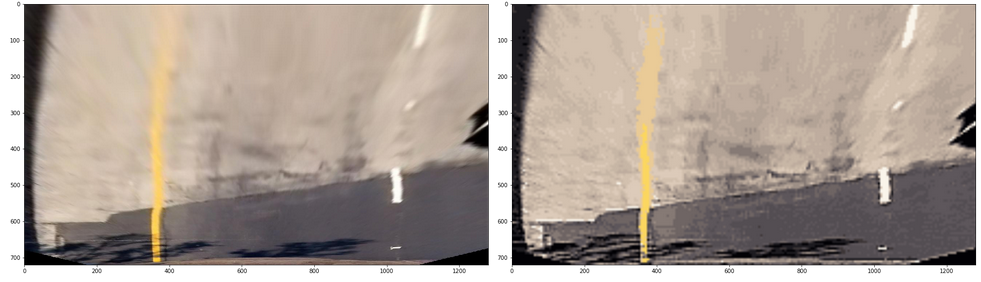
レーンは鳥瞰図の中に含まれる情報のうちユニークな明るい色を持ち、且つ縦方向に連なる特徴があること考えられます。

まず鳥瞰図に対してcv2.kmeans関数を用いて含まれるRGB情報を16パターンに減色する処理を加えた。

Now we will extract the lanes contained in the image from the bird's-eye view.

The lane is considered to have a unique bright color out of the information contained in the bird's-eye view, and features that are continuous in the vertical direction.

First we added a process to reduce the RGB information contained in the bird's-eye view using the cv2.kmeans function to 16 patterns.

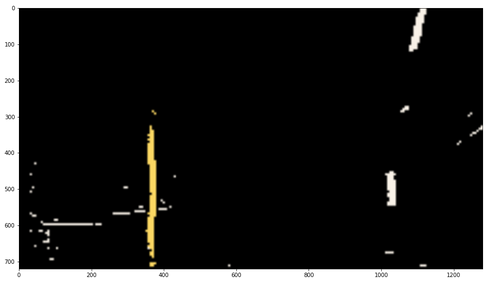


次に、残った16パターンの中から面積の占める割合が多いラベル、横方向に長いラベルを除外する。さらにラベルに含まれる各画素をグレースケール変換して、輝度が100pixel以下の画素を除外した。RGB値からグレースケール輝度への変換はG=R\*0.299+B\*0.587+R\*0.114式を用いた

結果、ユニークで明るい色を持ち、且つ縦方向に連なる特徴を持つラベルをレーンとして抽出した。

Next, from the remaining 16 patterns, labels that occupy a large proportion of the area and labels that are long in the horizontal direction are excluded. Further, each pixel included in the label was subjected to grayscale conversion to exclude pixels having luminance of 100 pixels or less. The conversion from RGB value to grayscale luminance was calculated using G = R \* 0.299 + B \* 0.587 + R \* 0.114

As a result, we extracted labels with unique and bright colors and features that are continuous in the vertical direction as lanes.

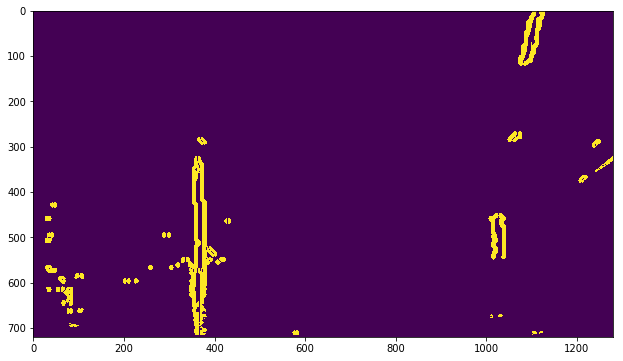


それでもまだ残った僅かな横方向のノイズ分を除去するためにX方向に対するSOBELを適用した。

最後に残った画素を２値化して、レーン検出に必要なバイナリイメージを作成した。

Nevertheless, SOBEL for the X direction was applied in order to remove a slight amount of lateral noise which remained.

The remaining pixels are binarized to create a binary image necessary for lane detection.



#### 4. Describe how (and identify where in your code) you identified lane-line pixels and fit their positions with a polynomial?

４−１　左右レーンを示す近似曲線

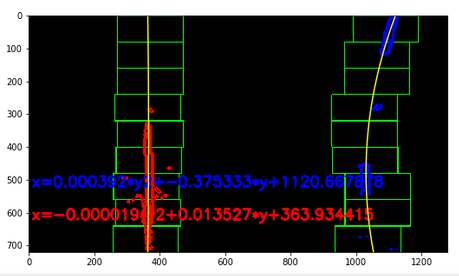
レーン検出に必要なバイナリイメージにスライディング法を使用して、ヒストグラムのピーク（レーンであろうと仮定する画素の塊）を基準に、「レーン抽出のための小エリアを縦方向」に連ねて、含まれる画素より表示レーンを２次方程式による近似曲線で示すこととした。

本サンプル画像の場合、結果右レーン（青）は式x=0.000392\*y^2+-0.375333\*y+1120.667878（単位ピクセル）、左レーン（赤）はx=-0.000019\*y^2+0.013527\*y+363.934415（単位ピクセル）の近似曲線として計算できた。

4-1 Approximation curve showing left and right lanes

Using the sliding method for the binary image necessary for lane detection, the "small area for lane extraction" is included in the "vertical direction for lane extraction" with reference to the peak of the histogram (the lump of pixels assumed to be a lane) The display lane is indicated by an approximate curve based on a quadratic equation rather than the pixel to be displayed.

In the case of this sample image, the right lane (blue) of the result is x = 0.000392 \* y ^ 2 + - 0.375333 \* y + 1120.667878 (unit pixels) and the left lane (red) is x = - 0.000019 \* y ^ 2 + 0.013527 \* It was calculated as an approximate curve of y + 363.934415 (unit pixels).

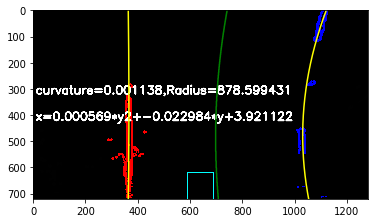


４−２　レーンを示す近似曲線（中央位置）

次に左右レーン毎に取得した２次方程式を用いて黄色線を描画した。その後左右の黄色線について、高さy位置に置ける横ｘ中心点を記録した。記録した中心点ｘ，ｙの集まりからレーン中央位置を示す２次方程式より得られた式x=0.000187\*y^2+-0.181181\*y+741.833852（単位ピクセル）で示す近似曲線（緑線）を「レーンを示す近似曲線」と定めた。

4-2 Approximate curve showing the lane (center position)

Next, yellow lines were drawn using quadratic equations obtained for each of the left and right lanes. After that, for the left and right yellow lines, the horizontal x center point that can be placed at the height y position was recorded. An approximate curve (green line) indicated by the equation x = 0.000187 \* y ^ 2 + - 0.181181 \* y + 741.833852 (unit pixel) obtained from a quadratic equation indicating the central position of the lane from the collected central points x, It is defined as "approximate curve showing lane".



**5. Describe how (and identify where in your code) you calculated the radius of curvature of the lane and the position of the vehicle with respect to center.**

５−１　車両のレーン逸脱距離

自車の先端はカメラ基準となるので画面下部中心四角枠位置（６４０，７２０）付近に位置すると仮定する。  
 緑線の下部ｘ位置はおよそ708ピクセルであり、その間のズレ１２ピクセルを逸脱距離と仮定する。

プロジェクト内の情報より、画像単位[ピクセル]と実空間単位[m]の変換率は以下のとおりである。

ym\_per\_pix = 30/720 # meters per pixel in y dimension  
xm\_per\_pix = 3.7/700 # meters per pixel in x dimension

車両とレーンのズレ１２ピクセルは実空間上で約0.0634mと推定することができる。

5-1 Lane departure distance of vehicle

It is assumed that the tip of the vehicle is located near the center of the screen lower center rectangular frame (640, 720) because it is the camera reference.

  The lower x position of the green line is approximately 708 pixels, and the deviation distance is assumed to be 12 pixels between them.

From the information in the project, the conversion rates of image unit [pixel] and real space unit [m] are as follows.

ym\_per\_pix = 30/720 # meters per pixel in y dimension

xm\_per\_pix = 3.7 / 700 # meters per pixel in x dimension

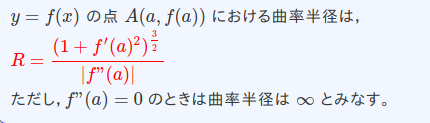
The displacement of 12 pixels between the vehicle and the lane can be estimated to be about 0.0634 m in the real space.

５−２　実際の道路曲率半径（勾配）

４−２でレーン中央位置を示す近似曲線を求める際に記録した左右黄色線の中心点情報ｘ,ｙ情報にxm\_per\_pix とym\_per\_pix を掛けて実空間上での位置距離情報X,Yを計算した。

得たX,Y位置情報に対して２次方程式で示すこととした。

その結果、x=0.000569\*y^2+-0.022984\*y+3.921122　（単位ｍ）による近似曲線を得た。

ところで道路曲率半径は以下式で定義できる。

x=f(ｙ）＝ 0.000569\*y^2+-0.022984\*y+3.921122の道路で、自車位置y=720ピクセルなので  
ｆ’(a) ＝2.＊0.000569＊（720＊ym\_per\_pix）-0.022984  
ｆ’’(a) ＝ 2.＊0.000569　であるから、　道路曲率半径は878.599431ｍと計算できた。

5-2 Actual road curvature radius (slope)

Position distance information X, Y on the real space was calculated by multiplying xm\_per\_pix and ym\_per\_pix with the center point information x, y information of the left and right yellow lines recorded at the time of obtaining the approximate curve indicating the center position of the lane at 4-2.

It is decided by quadratic equation with respect to the obtained X, Y position information.

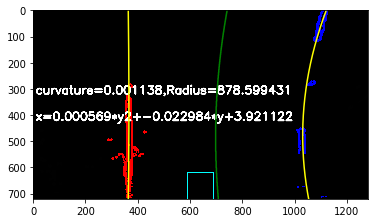
As a result, an approximate curve was obtained by x = 0.000569 \* y ^ 2 + - 0.022984 \* y + 3.921122 (unit m).

By the way, the road curvature radius can be defined by the following formula.

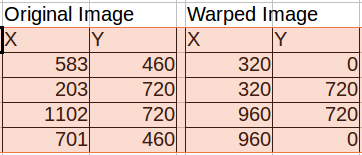
Since the vehicle position y = 720 pixels on the road of x = f (y) = 0.000569 \* y ^ 2 + - 0.022984 \* y + 3.921122

f '(a) = 2. \* 0.000569 \* (720 \* ym - per - pix) - 0.022984

Since f '' (a) = 2. \* 0.000569, the road curvature radius can be calculated as 878.599431 m.



#### 6. Provide an example image of your result plotted back down onto the road such that the lane area is identified clearly.



最後に私は鳥瞰図上で得た情報を元の基準画像に逆写像することに挑戦した。基準座標から鳥瞰図の変換で用いた4点の対応は右表のとおりであった。

そこでcv2.getPerspectiveTransform関数の引数に対して入力srcと出力dstを入れ替えた逆M行列を求めた。

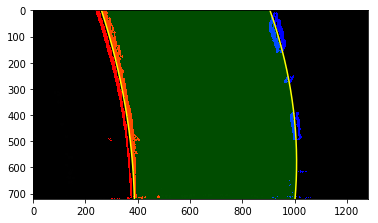
鳥瞰図上から基準画像上に逆写像するために私は逆M行列とcv2.perspectiveTransform関数を使用した。  
各左右レーンのX,Y位置を鳥瞰図上で取得し、基準画像に変換したものが下図の結果である。

Finally I tried to inverse map the information obtained on the bird's eye view to the original reference image. The correspondence of the four points used in the conversion of the bird's eye view from the reference coordinates was as shown in the right table.

Therefore, an inverse M matrix obtained by exchanging input src and output dst with respect to the argument of cv2.getPerspectiveTransform function was found.

I used an inverse M matrix and the cv2.perspectiveTransform function to reverse map from the bird's eye view onto the reference image.

The X and Y positions of each left and right lane are acquired on a bird's-eye view and converted to a reference image is the result of the following figure.



### Pipeline (video)

#### 1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (wobbly lines are ok but no catastrophic failures that would cause the car to drive off the road!).

以上の処理を組み合わせて作成した動画ファイルを提出する。  
開発したレーン検出器で自分の進むべきレーンを指し示す様子を動画ファイル上で確認できる。

Movie files created by combining the above processes are submitted.

You can check the motion picture file showing the lane that you should go with the lane detector you developed



### Discussion

#### 1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

・デッドレコニング機能の不足

左右レーン近似曲線の計算結果をそのまま利用したことで次の２つの課題が発生した。

①道路近似曲線（レーン中央）、道路曲率半径の値等は計算の度にブレて安定しない。

②左右いずれかのレーンが正しく検出できなかった場合、精度や結果に大きく得依拠する。

これらの対応としては、過去に求めたレーンの２次式係数、近似曲線、スライディングウィンドウの基点・開始位置を長期間記録し、もしもの際に補完／バックアップできるデッドレコニングのような機能を追加する必要がある。

· Lack of dead reckoning function

By using the calculation result of the right and left lane approximate curve as it is, the following two problems occurred.

① The road approximate curve (center of the lane), the value of the curvature radius of the road etc. are not stabilized at every calculation.

② If one of the left and right lanes can not be detected correctly, it depends largely on accuracy and results.

For these correspondences, record quadratic expression coefficients, approximate curves, sliding window base point / start position obtained in the past for a long time, and add functions like dead reckoning that can be supplemented / backed up in case There is a need.

・画像改質の強化

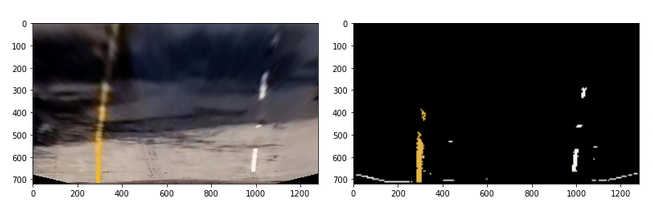
kmeanによる減色処理、あるいはHLSやエッジなどについては、解像度の不足する遠方や日影の影響でコントラストの確保が用意ではない（下図の場合、鳥瞰図上部のレーンが道路と同化してしまった)。

さらなる画像の事前の改質作業が望まれる。

· Strengthen image reform

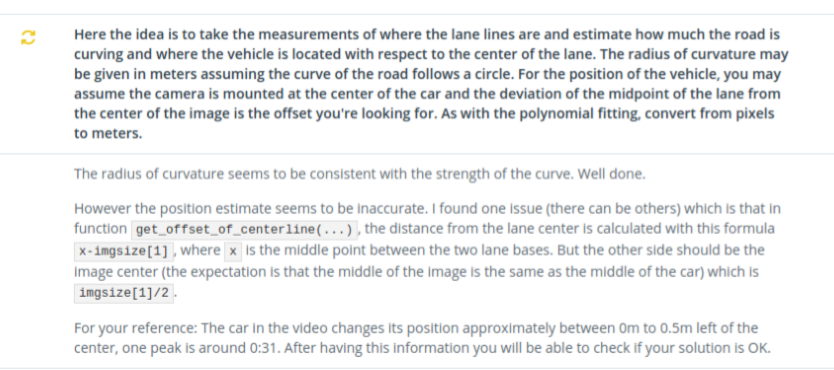
Regarding color reduction processing by kmean, HLS, edges, etc., it is not prepared to secure the contrast by the influence of shadows and shadows where the resolution is insufficient (In the case shown below, the lane at the top of the bird's-eye view has been assimilated with the road).

Advance reforming of further images is desired.



―――――――――――――――――――――――――――――――――――――――――――――――――――――――――――

[Error correction]



From the Mentor's comment, I recognized that there was an error in the calculation method of the road center.

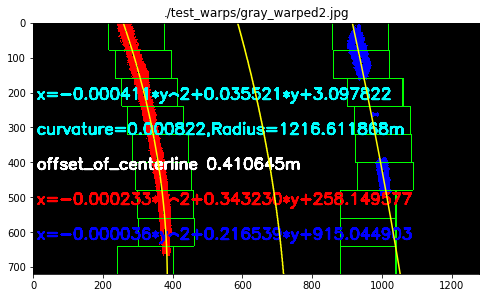
In order to get the difference between the correct road center and offset, I first modified "function: get\_offset\_of\_centerline".

However, I found a more serious mistake in "function: get\_list\_img\_sizeYX" My mistake is to define the inside and reverse width and height. As a result, I noticed that the conversion rate and image center were shifted. <Before correction> img = io.imread('./camera\_cal/calibration1.jpg') gbl\_img\_sizeX=img.shape[0] gbl\_img\_sizeY=img.shape[1]

<After>

img = io.imread('./camera\_cal/calibration1.jpg') gbl\_img\_sizeX=img.shape[1] gbl\_img\_sizeY=img.shape[0]

※I also modified other parts that use get\_list\_img\_sizeYX

By correcting these mistakes, I was able to keep the deviation from the center of the road at a predetermined level.

Additional problem countermeasures

In the Discussion, I mentioned about " dealing when lane can not be detected with good accuracy" and "dealing with detecting particles to obtain more lanes".

Although slightly, I have taken countermeasures for my two recommendations. As a result, erroneous detection of the running area could be avoided.

[Part\_A] . Added function to "dealing when lane can not be detected with good accuracy"

① Improvement of "the peak of histogram showing left and right lanes"

　→　　at　function of “ans\_project\_move\_function”　

The "peak of the histogram showing the left and right lanes" as the base point of the lowest slide is replaced with the calculated average value together with each of the left and right peaks detected the previous time to improve the accuracy

②Supplementation of detected particles

　→　　at　function of “ans\_project\_move\_function”　

In preparation for cases where binary particles required for lane detection can not be secured due to the influence of external light, the number of particles is supplemented using particles remaining in "binary\_warped" of the previous and last time.

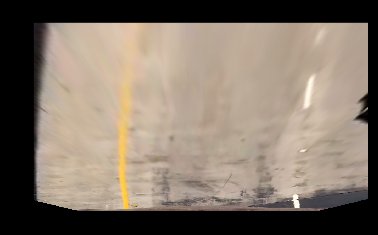
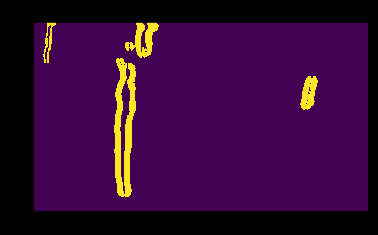
③Correction when one lane can not be detected

　→　　at function of “polynomial2ndfit”

When obtaining the quadratic approximate expression of the lane, when there is a difference of 50 times or more between the left and right binarized particles necessary for detection, in order to improve the reliability of the approximate expression, " Approximate coefficient of lane "is overwritten using the other (approximate coefficient of the lane in which more particles are detected).

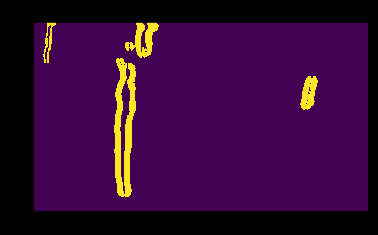
Areas where lane detection is difficult

[example: Before measures]

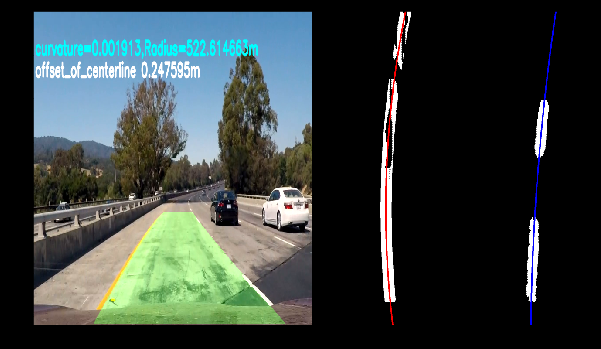
Input to　sliding　opration.

[example:After measures]: Complement with the previous results

Complemented lane-detect

Complemented lane-detect



[Part\_B] Added function to "dealing with detecting particles to obtain more lanes".

①　Enhancement of lane particle detection processing by kmean method

　→　　at function of “image\_decolor\_debug”

I added a method to combine the detection process of road lane particles by the kmean method which I have done so far in duplicate.

The added window was set at the top of the image where the resolution declined, and the parameters were adjusted so that the number of new labels was changed.

In addition, the image\_decolor\_debug function also adds a masking process to remove particles generated from bumpers displayed at the bottom of the screen.

[example: Before measures]

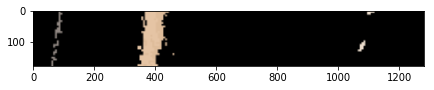
Exclude bumper display area

Areas where lane detection is difficult





Complemented lane-detect



Process the top of the screen with different tune-ups