ImageProcessing - intrinsic: cv::Mat - distortionCoeffs: cv::Mat - gaussianSigmaX: double - gaussianSigmaY: double - minThreshHLS: cv::Scalar - maxThreshHLS: cv::Scalar - minThreshBGR: cv::Scalar - maxThreshBGR: cv::Scalar + ImageProcessing(void): void + ~ImageProcessing(void): void + preProcessing(cv::Mat& src, cv::Mat& dst): void + getBinaryImg(cv::Mat& src, cv::Mat& dst): void + prespectiveTransform(cv::Mat& src, cv::Mat& dst, cv::Mat& T_perspective_inv): void + setIntrinsic(double fx_, double fy_, double cx_, double cy_): void + setDistCoeffs(double k1_, double k2_, double p1_, double p2_, double k3_): void + setgaussianSigmaX(double gaussianSigmaX_): void + setgaussianSigmaY(double gaussianSigmaY_): void + setMinThreshHLS(cv::Scalar minThreshHSL_): void + setMaxThreshHLS(cv::Scalar maxThreshHSL_): void + setMinThreshBGR(cv::Scalar minThreshBGR_): void + setMaxThreshBGR(cv::Scalar maxThreshBGR_): void + getIntrinsic(void): cv::Mat + getDistCoeffs(void): cv::Mat + getgaussianSigmaX(void): double + getgaussianSigmaY(void): double + getMinThreshHLS(void): cv::Scalar + getMaxThreshHLS(void): cv::Scalar + getMinThreshBGR(void): cv::Scalar + getMaxThreshBGR(void): cv::Scalar

LaneDetection

- windowBuffer: int
- avgRightCenter: std::vector<int>
- leftLaneCoeffs: cv::Mat
- rightLaneCoeffs: cv::Mat
- + LaneDetection(void): void
- + ~LaneDetection(void): void
- + generateHist(cv::Mat& src, std::vector<double>& hist): void
- + averageWindowCenter(int& xVal): int
- + extractLane(cv::Mat& perspectiveImg, std::vector<double>& hist, std::vector<cv::Point>& dstLane, std::string laneType): void
- + fitPoly(std::vector<cv::Point> laneLR, cv::Mat dstLaneParameters, int order): void
- + extractCentralLine(std::vector<cv::Point_<int>> leftLane, std::vector<cv::Point_<int>> rightLane, std::vector<cv::Point_<int>> centralLine): void
- + computeTurnAngle(std::vector<cv::Point>& leftLine): double
- + detectLanes(void): void
- + displayOutput(cv::Mat& src, cv::Mat& t_inv,LaneInfo& leftlane, LaneInfo& rightlane): void



LaneInfo

- lanePoints: std::vector<cv::Point>
- laneCoeffs: cv::Mat
- laneColor: cv::Vec3b
- + LaneInfo(void): void
- + ~LaneInfo(void): void
- + setLanePoints(std::vector<cv::Point> lanePoints): void
- + setLaneCoeffs(cv::Mat laneCoeffs): void
- + setLaneColor(cv::Vec3b laneColor): void
- + getLanePoints(void): std::vector<cv::Point>
- + getLaneCoeffs(void): cv::Mat
- + getLaneColor(void): cv::Vec3b