تشخیص حیوانات در جاده و جلوگیری از برخورد

Animal Detection and Collision Avoidance

دانشجویان : مسعود جانفشان ، ساناز الهیاری

استاد محترم : آقای دکتر سلطانی

دانشگاه آزاد مشهد

فهرست

مقدمه

مقاله با عنوان :

"A Practical **Animal Detection** and **Collision Avoidance** System Using Computer Vision Technique"

به موضوع **تشخیص حیوانات** و **جلوگیری از برخورد خودرو** با آنها در جاده‌ها و بزرگراه‌ها پرداخته است.

مسئله اصلی این مقاله، کاهش تصادفات جاده‌ای ناشی از برخورد خودروها با حیوانات است که منجر به آسیب‌های جانی و مالی می‌شود. این موضوع به‌ویژه در کشورهایی مانند هند که حیوانات به‌طور آزادانه در جاده‌ها تردد می‌کنند، بسیار حائز اهمیت است.

در بسیاری از موارد، رانندگان **به‌موقع** متوجه حضور حیوانات در جاده نمی‌شوند و زمان کافی برای واکنش و جلوگیری از برخورد را ندارند. این موضوع نیاز به توسعه سیستم‌های هوشمند تشخیص حیوانات و هشدار به راننده را بیش از پیش آشکار می‌کند.

هدف اصلی، توسعه یک سیستم **عملی** و **کم‌هزینه** است که بتواند حیوانات را در جاده تشخیص دهد و به راننده هشدار لازم را برای جلوگیری از برخورد بدهد.

اهمیت این مسئله در کاهش **تصادفات جاده ای** ، **حفاظت از حیوانات** و **افزایش ایمنی جاده‌ها** است

در این نوع سیستم با چالش هایی رو به رو هستیم :

1. **تنوع در شکل، اندازه و رفتار حیوانات :** حیوانات در شکل‌ها، اندازه‌ها و رفتارهای مختلفی ظاهر می‌شوند که تشخیص آنها را دشوار می‌کند.
2. **شرایط محیطی متغیر :** شرایط نوری و آب‌وهوایی مختلف (مانند نور کم، باران، مه و غیره) می‌تواند بر عملکرد سیستم تشخیص حیوانات تأثیر بگذارد.
3. **سرعت خودرو و حیوان :** سرعت خودرو و حیوان می‌تواند بر دقت تشخیص و زمان واکنش راننده تأثیر بگذارد. در سرعت‌های بالا، زمان کافی برای واکنش راننده وجود ندارد.
4. **هزینه و پیچیدگی سیستم :** توسعه سیستم‌های تشخیص حیوانات باید کم‌هزینه و عملی باشد تا بتواند به‌طور گسترده در خودروها و جاده‌ها استفاده شود.

روش پیشنهادی

در این مقاله ، روش اصلی برای تشخیص حیوانات و جلوگیری از برخورد خودرو با آنها، ترکیبی از دو تکنیک مهم در پردازش تصویر و بینایی ماشین است:

۱:Histogram of Oriented Gradients (HOG)

یک توصیف‌گر ویژگی (feature descriptor) است که برای تشخیص اشیا در تصاویر و ویدیوها استفاده می‌شود. این روش با تجزیه و تحلیل گرادیان‌های جهت‌دار در تصویر، ویژگی‌های حیوانات در تصاویر را را استخراج می‌کند.

۲:Cascade Classifier

یک روش یادگیری ماشین است که برای تشخیص اشیا استفاده می‌شود. این روش از چندین طبقه‌بند (classifier) تشکیل شده که به صورت آبشاری (cascade) عمل می‌کنند.

اما روش پیاده سازی کد متفاوت است :

این کد بر اساس یک روش **تشخیص اشیا مبتنی بر یادگیری عمیق (YOLOv5)** پیاده سازی شده است، که متفاوت از روش ذکر شده در مقاله (HOG و Cascade Classifier) است، که در ادامه به آن خواهیم پرداخت.

مقاله چه نوآوری یا مزیتی نسبت به روشهای قبلی ارائه میدهد؟

روش‌های قبلی تشخیص حیوانات معمولاً محدودیت‌هایی مانند دقت پایین، هزینه بالا، وابستگی به شرایط محیطی، و عدم توانایی کار در محیط‌های پویا و شلوغ (مانند جاده‌ها) داشته‌اند.

اما در این مقاله تمرکز بر ارائه یک سیستم **عملی** و در عین حال **کم هزینه** بوده که از ترکیب HOG و Cascade Classifier استفاده می‌کند و سعی کرده است این محدودیت‌ها را برطرف کند. این سیستم می‌تواند حیوانات را با دقت نسبتاً بالا تشخیص دهد و به راننده هشدارهای لازم را برای جلوگیری از برخورد بدهد. مهم‌ترین مزایا و نوآوری ها :

استفاده از ترکیب HOG وCascade Classifier

- محاسبه فاصله حیوان از خودرو
- سیستم هشدار بر اساس فاصله و سرعت
- تمرکز بر شرایط جاده
- دقت قابل قبول با هزینه کم

- آزمایش در شرایط واقعی
- قابلیت توسعه برای حیوانات دیگر(مقاله به‌طور خاص برای تشخیص سگ و گاو در جاده های هند طراحی شده اما میتوان آن را برای تشخیص سایر حیوانات نیز توسعه داد)

در این مقاله ، از روش‌های **کلاسیک** پردازش تصویر و بینایی ماشین استفاده شده است، در ادامه به ویژگی های این روش اشاره میکنیم:

- **هزینه پایین** : روش‌های کلاسیک نیازی به سخت افزارهای گران‌قیمت (مانند GPU) ندارند و می‌توانند روی سیستم‌های معمولی اجرا شوند.
- **سادگی پیاده‌سازی** : روش‌های کلاسیک مانند HOG و Cascade Classifier ساده‌تر هستند و پیاده‌سازی آنها آسان‌تر است.
- **نیاز به داده‌های آموزشی کمتر** :روش‌های کلاسیک مانند HOG نیازی به مجموعه‌داده‌های بزرگ برای آموزش ندارند و می‌توانند با داده‌های محدود کار کنند.
- **سرعت بالا** :روش‌های کلاسیک مانند Cascade Classifier می‌توانند در زمان واقعی کار کنند و برای سیستم‌های بلادرنگ مناسب هستند.

مقاله از **ترکیب** دو روش کلاسیک پردازش تصویر، یعنی **HOG** و **Cascade Classifier**، برای تشخیص حیوانات استفاده می کند به این گونه که ویژگی های تصاویر استخراج شده و در نهایت طبقه بندی می شوند

HOG به عنوان توصیف‌گر ویژگی:

- HOG ویژگی‌های حیوانات را از تصویر استخراج می‌کند. این ویژگی ها شامل اطلاعاتی درباره شکل ، لبه‌ها ، و جهت‌گیری حیوانات است.
- این ویژگی‌ها به‌عنوان ورودی به Cascade Classifier داده می‌شوند.

Cascade Classifier به عنوان طبقه‌بند:

- Cascade Classifierاز ویژگی‌های استخراج‌شده توسط HOG استفاده می‌کند تا حیوانات را تشخیص دهد.
- این طبقه‌بند به‌صورت آبشاری عمل می‌کند و در هر مرحله، پنجره‌هایی که احتمالاً حاوی حیوان نیستند، حذف می‌شوند.

این ترکیب باعث می‌شود سیستم بتواند در زمان واقعی کار کند و دقت قابل قبولی (۸۲.۵٪) داشته باشد.

مجموعه داده ها(Datasets)

در مقاله از مجموعه داده‌های مخصوص خود استفاده کرده است که شامل انواع تصاویر حیوانات هستند .

این مجموعه به دو نوع از تصاویر تقسیم میشود :

تصاویر مثبت : این تصاویر شامل حیوانات هستند که سیستم باید آنها را تشخیص دهد.(تصاویر حیوانات از گونه‌های مختلفی که در جاده ها یافت می‌شوند)

تصاویر منفی : این تصاویر شامل صحنه‌هایی بدون حیوانات هستند که سیستم باید آنها را نادیده بگیرد.(صحنه‌های جاده‌ها و بزرگراه‌ها که شامل ماشین‌ها، درختان، ساختمان‌ها و سایر اشیاء غیرحیوانی هستند)

- بیش از **۲۲۰۰ تصویر** برای آموزش سیستم استفاده شده است که شامل **۷۰۰ تصویر مثبت** و **۱۵۰۰ تصویر منفی** است

مجموعه داده‌ها برای آزمایش در مقاله:

از ویدیوهای واقعی برای آزمایش سیستم استفاده شده است . این ویدیوها شامل صحنه‌هایی از جاده‌ها و بزرگراه‌های هند هستند که حیوانات (به‌ویژه گاو) در آنها حضور دارند.

ویدیوها با استفاده از یک دوربین معمولی که روی خودرو نصب شده است، ضبط شده‌اند.

این ویدیوها شامل شرایط مختلفی مانند نور روز، نور کم، و شرایط آب‌وهوایی مختلف (مانند باران و مه) هستند.

ویدیوها با سرعت‌های مختلف خودرو (از ۰ تا ۶۰ کیلومتر بر ساعت) آزمایش شده‌اند.

سیستم در شرایط مختلفی مانند فاصله حیوان از خودرو، سرعت خودرو، و شرایط نوری مختلف آزمایش شده است.

مجموعه داده های جایگزین برای آموزشی:

در کد از مدل **YOLOv5** استفاده شده که این مدل یک مدل **مبتنی بر یادگیری عمیق** است که از **شبکه های کانولوشنال (CNN)** استفاده میکند. این مدل روی یک مجموعه داده ی بزرگ **عمومی** مانند **COCO** آموزش داده شده است.

مجموعه داده‌ها جایگزین برای آزمایش:

- این ویدیو از ترکیب چندین ویدیوی ظبط شده ی واقعی از رانندگی در جاده با شرایط مختلف و حیوانات گوناگون تشکیل شده است.

مجموعه‌داده‌ی COCO :

COCO (Common Objects in Context) یک مجموعه داده‌ی بزرگ و معتبر است که شامل ۸۰ کلاس مختلف از اشیا است و از ۲۰۰ هزار تصویر با برچسب های دقیق استفاده کرده.

کلاس های حیوانات در COCO شامل گاو - سگ - گربه - اسب - گوسفند و غیره است

نحوه پیش‌پردازش داده‌ها:

- در روش **مقاله** تصویر رنگی به تصویر خاکستری تبدیل می‌شود و نور و کنتراست آن تنظیم می‌شود و در نهایت هیستوگرام گرادیان هایشان استخراج شده و طبقه بندی میشوند

- در روش کد تصاویر به اندازه مشخصی (1280 x 720) پیکسل تغییر اندازه داده می‌شود و برای اینکه به عنوان ورودی به YOLOv5 داده شود به تنسور تبدیل میشوند و در نهایت در مدل YOLOv5 پیکسل ها در تصاویر به محدوده ای بین 0 تا 1 نرمال سازی می شوند .

نتایج و مقایسه عملکرد

در مقاله از معیارهای استاندارد مانند **دقت، حساسیت، اختصاصیت، نرخ مثبت کاذب، نرخ منفی کاذب، منحنی ROC، زمان پردازش، فاصله تشخیص، سرعت خودرو و تعداد فریم‌های آزمایشی** برای ارزیابی عملکرد سیستم استفاده شده است. این معیارها به‌طور جامع عملکرد سیستم را از نظر دقت تشخیص، سرعت، و قابلیت استفاده در شرایط واقعی ارزیابی می‌کنند .

مقایسه نتایج مقاله با روش های قبلی :

1) **مقایسه با روش‌های مبتنی بر حرکت(Motion Detection)**

- روش قبلی :در روش‌های مبتنی بر حرکت، فرض می‌شود که پس‌زمینه ثابت است و هر شیء متحرکی به عنوان حیوان تشخیص داده می‌شود.
- محدودیت‌ها :این روش در محیط‌های پویا (مانند جاده‌ها) دقت پایینی دارد و ممکن است اشیاء دیگر به اشتباه به عنوان حیوان تشخیص داده شوند.
- پیشرفت سیستم پیشنهادی :سیستم پیشنهادی از ترکیب HOG و Cascade Classifier استفاده می‌کند که دقت تشخیص را افزایش داده و مثبت‌های کاذب را کاهش می‌دهد.

2) **مقایسه با روش‌های مبتنی بر آستانه‌گذاری(Threshold Segmentation)**

- روش قبلی :در این روش، یک آستانه برای جدا کردن حیوان از پس‌زمینه تعیین می‌شود.
- محدودیت‌ها :انتخاب آستانه مناسب دشوار است و این روش در شرایط نوری مختلف عملکرد ضعیفی دارد.
- پیشرفت سیستم پیشنهادی :سیستم پیشنهادی از HOG برای استخراج ویژگی‌ها استفاده می‌کند که نسبت به تغییرات نور و کنتراست مقاوم‌تر است.

3) **مقایسه با روش‌های مبتنی بر تشخیص چهره(Face Detection)**

- روش قبلی :در این روش، چهره حیوانات تشخیص داده می‌شود.
- محدودیت‌ها :این روش نیاز دارد که حیوان به دوربین نگاه کند و برای حیوانات با شکل‌ها و اندازه‌های مختلف دقت پایینی دارد.
- پیشرفت سیستم پیشنهادی :سیستم پیشنهادی از HOG و Cascade Classifier استفاده می‌کند که می‌تواند حیوانات را از جهات مختلف و در اندازه‌های مختلف تشخیص دهد.

4) مقایسه با روش‌های مبتنی بر توصیف‌گرهای بافت(Texture Descriptors)

- روش قبلی :در این روش، از توصیف‌گرهای بافت (مانندSIFT) برای تشخیص حیوانات استفاده می‌شود.
- محدودیت‌ها :این روش فقط برای ویدیوهایی با یک حیوان و پس‌زمینه ساده کار می‌کند و در محیط‌های شلوغ دقت پایینی دارد.
- پیشرفت سیستم پیشنهادی :سیستم پیشنهادی می‌تواند حیوانات را در محیط‌های شلوغ و با پس‌زمینه‌های پیچیده تشخیص دهد.

5) مقایسه با روش‌های مبتنی بر GPS وGPRS

- روش قبلی :در برخی سیستم‌ها از GPS و GPRS برای ردیابی حیوانات استفاده می‌شود.
- محدودیت‌ها :این سیستم‌ها هزینه بالایی دارند و به دلیل وابستگی به پارامترهای مختلف (مانند سرعت حیوان و تأخیر در دریافت پیام) ممکن است خطاهای زیادی داشته باشند.
- پیشرفت سیستم پیشنهادی :سیستم پیشنهادی کم‌هزینه است و نیازی به سخت‌افزارهای گران‌قیمت ندارد.

6) مقایسه سیستم پیشنهادی روش (HOG + Cascade Classifier) و روشHaar Cascade

- سیستم پیشنهادی دقت و سرعت بالاتری نسبت به Haar Cascade دارد. منحنی ROC سیستم پیشنهادی نیز عملکرد بهتری را نشان می‌دهد.

جمع‌بندی:

مقاله نشان می‌دهد که سیستم پیشنهادی ترکیب HOG و Cascade Classifier نسبت به روش‌های قبلی پیشرفت‌هایی در زمینه‌های زیر داشته است:

- ✓ **دقت تشخیص :دقت سیستم ٪۸۲.۵ است که نسبت به روش‌های قبلی بهبود قابل‌توجهی دارد.**
- ✓ **سرعت پردازش :سیستم می‌تواند در زمان واقعی کار کند و برای سیستم‌های بلادرنگ مناسب است.**
- ✓ **فاصله تشخیص :سیستم می‌تواند حیوانات را تا فاصله ۲۰ متری تشخیص دهد.**
- ✓ **هزینه کم :سیستم پیشنهادی کم‌هزینه است و می‌تواند به‌طور گسترده استفاده شود**

نقاط قوت روش پیشنهادی:

1 .دقت قابل قبول:

- ✓ سیستم پیشنهادی با دقت ٪۸۲.۵ حیوانات را تشخیص می‌دهد که نسبت به روش‌های قبلی بهبود قابل‌توجهی دارد.
- ✓ این دقت برای یک سیستم کم‌هزینه و عملی بسیار مناسب است.

2 .سرعت بالا:

- ✓ زمان پردازش سیستم ۱۰۰ میلی‌ثانیه (معادل ۱۰ فریم بر ثانیه) است که برای سیستم‌های بلادرنگ مناسب است.
- ✓ این سرعت به راننده زمان کافی برای واکنش می‌دهد.

3 .فاصله تشخیص مناسب:

- ✓ سیستم می‌تواند حیوانات را تا فاصله ۲۰ متری تشخیص دهد.
- ✓ این فاصله به راننده اجازه می‌دهد تا در صورت نزدیک شدن حیوان، هشدارهای لازم را دریافت کند.

4 .هزینه کم:

- ✓ سیستم پیشنهادی از دوربین‌های معمولی و نرم افزارهای متن باز(مانندOpenCV) استفاده می‌کند.
- ✓ این سیستم نیازی به سخت‌افزارهای گران‌قیمت(مانندLiDAR) یا رادار ندارد.

5 .سادگی پیاده‌سازی:

- ✓ روش‌های HOG و Cascade Classifier ساده هستند و پیاده سازی آنها آسان‌تر است.
- ✓ این سیستم می‌تواند به‌راحتی در خودروها و جاده‌ها استفاده شود.

6. مقاومت در برابر تغییرات نوری:

- ✓ HOGویژگی‌های حیوانات را به خوبی استخراج می‌کند و نسبت به تغییرات نور و کنتراست مقاوم‌تر است.
- ✓ این سیستم می‌تواند در شرایط نوری مختلف (مانند روز، شب، و شرایط آب‌وهوایی متفاوت) عملکرد نسبتاً خوبی داشته باشد.

۷ .کاهش مثبت‌های کاذب:

- ✓ ترکیب HOG و Cascade Classifier باعث می‌شود که مثبت‌های کاذب کاهش یابند و دقت سیستم افزایش پیدا کند.

نقاط ضعف روش پیشنهادی:

1 .محدودیت در سرعت خودرو:

- سیستم می‌تواند تا سرعت ۳۵ کیلومتر بر ساعت به راننده زمان کافی برای واکنش بدهد.
- در سرعت‌های بالاتر، اگرچه حیوان تشخیص داده می‌شود، اما زمان کافی برای جلوگیری از برخورد وجود ندارد.

2 .محدودیت در فاصله تشخیص:

- سیستم می‌تواند حیوانات را تا فاصله ۲۰ متری تشخیص دهد.
- برای افزایش ایمنی، نیاز به افزایش فاصله تشخیص وجود دارد.

3 .وابستگی به شرایط محیطی:

- اگرچه سیستم نسبت به تغییرات نور مقاوم‌تر است، اما در شرایط آب‌وهوایی بسیار نامساعد (مانند باران شدید یا مه غلیظ) ممکن است عملکرد آن کاهش یابد.

4 .عدم تشخیص در شب:

- سیستم پیشنهادی برای تشخیص حیوانات در شب طراحی نشده است.
- این یک محدودیت بزرگ است، زیرا بسیاری از تصادفات با حیوانات در شب اتفاق می‌افتد.

5 .نیاز به آموزش مجدد برای حیوانات دیگر:

- سیستم به‌طور خاص برای تشخیص گاو و سگ بهینه‌سازی شده است.
- برای تشخیص حیوانات دیگر (مانند گوزن، خوک، و غیره)، نیاز به آموزش مجدد و جمع‌آوری داده‌های جدید وجود دارد.

6 .محدودیت در تشخیص حیوانات کوچک:

- سیستم ممکن است در تشخیص حیوانات کوچک (مانند سگ‌های کوچک یا گربه‌ها) دقت کمتری داشته باشد.

۷ .وابستگی به کیفیت دوربین:

- عملکرد سیستم به کیفیت دوربین و وضوح تصویر وابسته است.
- دوربین‌های با کیفیت پایین ممکن است دقت سیستم را کاهش دهند

جمع بندی مقاله

مقاله به این نتیجه می‌رسد که سیستم پیشنهادی ترکیب HOG و Cascade Classifier یک راه‌حل عملی و کم‌هزینه برای تشخیص حیوانات و جلوگیری از برخورد خودروها با آنها است. این سیستم با دقت ٪۸۲.۵، سرعت پردازش ۱۰۰ میلی‌ثانیه، و فاصله تشخیص ۲۰ متری، می‌تواند به‌طور مؤثری از تصادفات جاده‌ای جلوگیری کند. با این حال، محدودیت‌هایی مانند محدودیت در سرعت خودرو، عدم تشخیص در شب، و نیاز به افزایش فاصله تشخیص وجود دارد که نیاز به توسعه بیشتر دارند.

***در مقاله پیشنهاداتی برای تحقیقات آینده ارائه شده است**

افزایش فاصله تشخیص:

بهبود سیستم برای تشخیص حیوانات در فاصله‌های دورتر (بیشتر از ۲۰ متر).

افزودن قابلیت تشخیص در شب:

توسعه سیستم برای تشخیص حیوانات در شرایط کم‌نور یا شب.

بهبود عملکرد در سرعت‌های بالاتر:

افزایش کارایی سیستم در سرعت‌های بالاتر از ۳۵ کیلومتر بر ساعت.

تشخیص حیوانات کوچک:

بهبود دقت سیستم در تشخیص حیوانات کوچک (مانند سگ‌های کوچک یا گربه‌ها).

استفاده از یادگیری عمیق:

جایگزینی روش‌های کلاسیک با الگوریتم‌های یادگیری عمیق (مانند YOLO یا SSD) برای دقت و سرعت بالاتر.

ادغام با سیستم‌های دیگر:

ترکیب سیستم تشخیص حیوانات با سیستم‌های ایمنی خودرو (مانند تشخیص عابر پیاده یا ترمز اضطراری).

استفاده از داده‌های بیشتر و متنوع‌تر:

جمع‌آوری و استفاده از داده‌های آموزشی بیشتر و متنوع‌تر برای بهبود دقت سیستم.

بهبود عملکرد در شرایط آب‌وهوایی نامساعد:

افزایش مقاومت سیستم در شرایطی مانند باران شدید، مه، یا برف.

توسعه سیستم برای حیوانات دیگر:

آموزش سیستم برای تشخیص انواع دیگر حیوانات (مانند گوزن، خوک، و غیره).

کاهش هزینه‌ها:

کاهش بیشتر هزینه‌های سیستم برای استفاده گسترده‌تر، به‌ویژه در کشورهای در حال توسعه

تحلیل عملکرد

پارامتر های استفاده شده در کد با پارامتر های استفاده شده در مقاله متفاوت است چون روش های به کار رفته متفاوت است

روش تشخیص اشیا

مقاله :از ترکیب HOG و Cascade Classifier استفاده می‌کند.

کد :از YOLOv5 استفاده می‌کند که یک روش مبتنی بر یادگیری عمیق است.

پارامترهای آموزش

مقاله: تعداد مراحل :۲۰

نوع ویژگی :HOG

ابعاد نمونه :۴۰x۷۰ پیکسل

نرخ تشخیص حداقل (minHitRate) :۰.۹۹۵

نرخ هشدار غلط حداقل (minFalseAlarmRate) :۰.۵

کد:

1. **نرخ یادگیری :(Learning Rate)** سرعت یادگیری مدل را کنترل می‌کند.
2. **تعداد دوره‌ها :(Epochs)** تعداد دفعاتی که مدل کل داده‌ها را می‌بیند.
3. **اندازه دسته :(Batch Size)** تعداد عکس‌هایی که در هر مرحله به مدل داده می‌شود.

پارامترهای تشخیص

مقاله: دقت سیستم: ٪۸۲.۵

فاصله تشخیص: تا ۲۰ متر

سرعت پردازش: ۱۰۰ میلی‌ثانیه (۱۰ فریم بر ثانیه)

کد: دقت و فاصله تشخیص به مدل YOLOv5 و تنظیمات آن بستگی دارد.

*سرعت پردازش به سخت‌افزار و اندازه فریم‌ها بستگی دارد.

پارامترهای هشدار

مقاله: هشدارها بر اساس فاصله حیوان از خودرو و سرعت خودرو ارائه می‌شوند.

کد: هشدارها بر اساس تعداد خطوط خطر فرضی که حیوان از آنها عبور می‌کند، ارائه می‌شوند.

منطقه خطر (ROI)

مقاله : منطقه خطر به‌صورت دستی تعریف نشده است.

کد : یک منطقه چندضلعی (polygonal ROI) تعریف شده است تا با ورود حیوان به آن منطقه هشدار صادر شود .

کلاس‌های مجاز

مقاله : فقط حیوانات (به‌ویژه گاو) تشخیص داده می‌شوند.

کد :از لیست **allowed_classes** استفاده کرده‌ایم که شامل انسان و چندین نوع حیوان است.

محاسبه فاصله

مقاله : محاسبه فاصله ارتفاع پایین ترین پیکسل کادر حیوان تا پایینترین نقطه تصویر (Y=0)

کد : **نسبت** (عرض واقعی سوژه x فاصله کانونی) نسبت به عرض کادر در تصویر

جمع‌بندی:

مقاله : از پارامترهای خاصی مانند HOG و Cascade Classifier با تنظیمات دقیق استفاده کرده است.

کد : از مدل YOLOv5 استفاده می‌کند که پارامترهای آن از پیش تنظیم شده‌اند و با پارامترهای مقاله متفاوت هستند

***مجموعه داده های مقاله با مجموعه داده های کد متفاوت است.**

- **مقاله** از یک مجموعه‌داده‌ی اختصاصی شامل بیش از ۲۲۰۰ تصویر (۷۰۰ تصویر مثبت و ۱۵۰۰ تصویر منفی) استفاده کرده است.

  این مجموعه‌داده شامل تصاویر حیوانات (به‌ویژه گاو) و صحنه‌های جاده‌ای بدون حیوانات است.

  مجموعه‌داده‌ی مقاله برای آموزش سیستم مبتنی بر **HOG**و **Cascade Classifier**استفاده شده است.

- **کد** از یک مدل از پیش آموزش‌دیده‌ی **YOLOv5** استفاده می‌کند.
  این مدل روی مجموعه‌داده‌ی **COCO** آموزش داده شده است.
  مجموعه داده‌ی COCO شامل ۸۰ کلاس مختلف از اشیا (مانند انسان، ماشین، حیوانات، و غیره) است.

ابزار های استفاده شده

زبان برنامه نویسی : Python

کتابخانه ها :

OpenCV : برای پردازش بر روی تصاویر و فریم ها و کار با ویدیو

Torch (PyTorch) : برای بارگذاری مدل اموزش یادگیری عمیق

Numpy : برای انجام محاسبات عددی در کد

ابزار های جانبی : Matplotlib و VideoWriter

چالش های پیاده سازی

در مقاله، روش و مدل به‌طور کامل توضیح داده شده بود، اما با توجه به عدم دسترسی به **کد اصلی مقاله** و **همچنین دیتاست استفاده‌شده** در آن، پیاده‌سازی دقیق این روش با چالش‌هایی مواجه بود. علاوه بر این، پیدا کردن یک دیتاست ساختاریافته که شامل تصاویر حیوانات با برچسب‌های دقیق باشد، کار را دشوارتر می‌کرد. این محدودیت‌ها باعث شدند که پیاده‌سازی روش مقاله به‌طور کامل سخت و زمان‌بر باشد.

در مقاله برخی جزئیات فنی مانند **پارامترهای دقیق آموزش مدل** و **روش کالیبراسیون دوربین** به‌طور کامل توضیح داده نشده بود. برای حل این چالش‌ها، از **مدل‌های از پیش آموزش‌دیده** مانند (YOLOv5) استفاده شد. این مدل‌ها به‌دلیل داشتن دقت بالا و سهولت در پیاده‌سازی، جایگزین مناسبی بودند. البته سبک و روش پیاده‌سازی در این حالت کاملاً متفاوت از روش مقاله است، اما نتیجه‌ی نهایی (تشخیص حیوانات و جلوگیری از برخورد) به‌خوبی محقق شد.

استفاده از مدل‌های از پیش آموزش‌دیده نه‌تنها چالش‌های پیاده‌سازی را کاهش داد، بلکه امکان **انعطاف‌پذیری بیشتر** در تشخیص انواع حیوانات و اشیا (به اضافه انسان) را نیز فراهم کرد. این رویکرد باعث شد تا سیستم به‌راحتی روی سخت‌افزارهای مختلف قابل اجرا باشد و نیاز به منابع محاسباتی کم‌تری داشته باشد.

دقت و نتیجه پیاده سازی به روش YOLOv5

با توجه به محدودیت هایی برای انجام تست ، مراحل تست فقط بر روی مجموعه ای از داده ها که شامل تعداد 632 نمونه **مثبت** و **منفی** بودند انجام شد و نتایج زیر حاصل گردید : (354 نمونه مثبت و 278 نمونه منفی)

صحیح مثبت (TP) : 299 ← تعداد حیواناتی که درست تشخیص داده شده‌اند.

صحیح منفی (TN) : 276 ← تعداد تصاویری که درست تشخیص داده شده‌اند که حیوان ندارند.

ناصحیح مثبت (FP) : 2 ← تعداد تصاویری که به‌اشتباه به‌عنوان حیوان تشخیص داده شده‌اند.

ناصحیح منفی (FN) : 55 ← تعداد حیوانات‌ای که تشخیص داده نشده‌اند.

در نهایت با محاسبه این دقت میتوان دقت کلی این سیستم را حدودا 91% اعلام کرد که در مقایسه با روش مقاله حدود 8 درصد دقت بیشتری دریافت می شود . البته که تست اصلی باید در شرایط واقعی در شرایط مناسب انجام شود .

عوامل تاثیر گذار در پیشرفت دقت سیستم نسبت به روش ارائه شده در مقاله

- **استفاده از مدل پیشرفته‌تر(YOLOv5) :**

YOLOv5یک مدل یادگیری عمیق است که نسبت به روش‌های قدیمی‌تر مثل (HOG + Cascade) دقت بالاتری دارد.

این مدل توانایی یادگیری ویژگی‌های پیچیده‌تر را دارد و بهتر می‌تواند حیوانات را در شرایط مختلف تشخیص دهد.

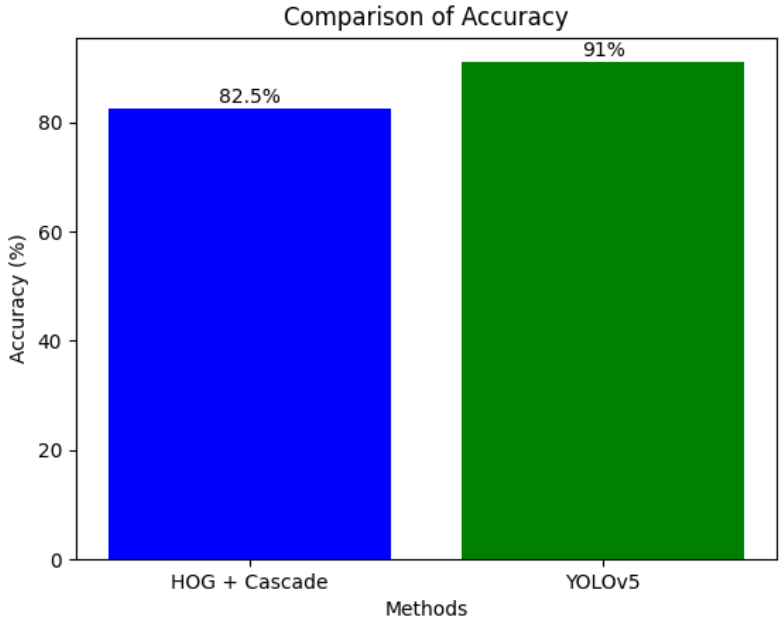
- **دیتاست بزرگ‌تر و متنوع‌تر:**

اگر از یک دیتاست بزرگ‌تر و متنوع‌تر برای آموزش یا تست شود می‌تواند دقت مدل را افزایش دهد.

دیتاست‌های بزرگ‌تر به مدل کمک کرده تا بهتر یاد بگیرد و در شرایط مختلف عملکرد بهتری از خود نشان بدهد

در ادامه عواملی از جمله موضوعات زیر می تواند تاثیر گذار باشد :

- پیش‌پردازش بهتر داده‌ها
- تنظیم پارامترهای بهینه
- سخت‌افزار بهتر



نحوه اجرا و عملکر کد

- مدل **YOLOv5** از یک فایل وزن از پیش آموزش دیده **(PreTrained)** در برنامه بارگذاری میشود و مدل اماده ی اجرا بر روی فریم های ویدیو مورد نظر می شود.
- ویدیوی مورد نظر با کمک ابزاری از **Cv2** خوانده می شود و تا فریم ها و اطلاعات ویدیو دریافت شوند . همچنین متغیری برای ویدیوی پردازش داده شده اختصاص داده می شود.
- هر فریم از ویدیو خوانده و پیش پردازش می شود (فریم ها به ابعاد **720x1280** تبدیل میشوند)
- مدل **YOLOv5** بر روی هر فریم اجرا میشود تا خروجی را تولید کند . این خروجی شامل اطلاعات مختصات شیء مورد نظر در تصویر و کلاس آن و میزان اطمینان (**Confidence**) است .
- کلاس اشیاء تشخیص داده شده فیلتر می شود تا تشخیص فقط مختص حیوانات و انسان باشد.
- با توجه به نسبت عرض واقعی حیوان تشخیص داده شده به عرض آن در تصویر فاصله اش در دوربین تخمین زده می شود .
- محدوده ی خطری فرضی در تصویر دوربین به نمایش در میاد و با توجه به این محدوده اگر حیوانی وارد این محدوده شود هشداری بر اساس فاصله آن به راننده صادر میشود و با توجه به این فاصله دستور ترمز هم نیز صادر میشود .
- بعد از اتمام پردازش بر روی تمام ویدیوی اولیه ، ویدیو پردازش شده ذخیره می شود .

توضیح کد

```
model_path = 'D:/MabaniBinaei/Project/ModelYolov5/ModelYolov5/yolov5s.pt'
```

برای شروع آدرس مدل YOLOv5s.pt که از پیش تعریف شده است را مشخص کرده و به سیستم وارد میکنیم.

```
device = "cpu"  # 0 For GPU

model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_path)

model.to(device)
```

در اینجا شیء تحت عنوان device تعریف می شود تا مقادیر آن نشان دهنده ی سخت افزار مورد نظر برای پردازش های مدل باشد و با دستور model.to(device) مدل به سخت افزار مورد نظر که در اینجا CPU هست داده شود . این کار باعث میشود تا محاسبات مدل روی CPU انجام شود .

با کمک تابع torch.hub.load مدل بارگزاری می شود .

- 'ultralytics/yolov5' نشان‌دهنده‌ی مخزن YOLOv5 (Repository) در GitHub است.
- 'custom' نشان می‌دهد که مدل از یک فایل وزن موجود در حافظه سیستم (Custom Model) بارگذاری می‌شود
- path=model_path مسیر فایل وزن مدل (yolov5s.pt) را مشخص می‌کند.

```
video = cv2.VideoCapture('D:/MabaniBinaei/Project/ModelYolov5/ModelYolov5
/finaltest2.mp4')

width, height = 1280, 720
fps = video.get(cv2.CAP_PROP_FPS)

_output1 = cv2.VideoWriter('output.mp4', cv2.VideoWriter_fourcc(*'mp4v'), fps, (width,
height)
```

ویدیو به کمک cv2.VideoCapture بارگذاری می شود و پارامتر آن آدرس ویدیو ی مد نظر است .

عرض (width) و ارتفاع (height) مورد نظر برای فریم‌های ویدیوی خروجی مشخص می شوند .

با دستور video.get فریم بر ثانیه (FPS) ویدیوی وارد شده مشخص میشود تا در پردازش ویدیوی خروجی مقدار آن به عنوان پارامتر برای ویدیویی خروجی تنظیم شود .

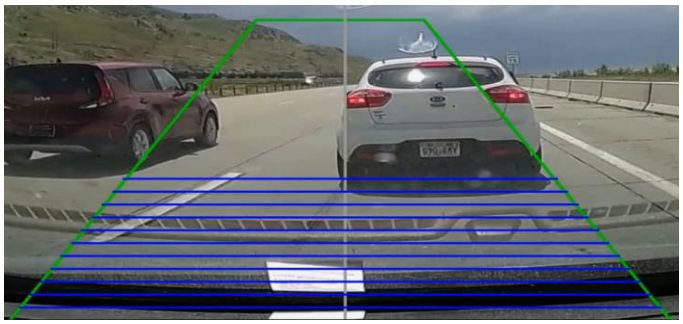
دستور اخر یک شیء VideoWriter ایجاد می‌کند تا ویدیوی پردازش‌شده را ذخیره کند. پارامترهای این تابع:

- 'output.mp4' نام فایل خروجی که ویدیوی پردازش‌شده در آن ذخیره می‌شود.
- cv2.VideoWriter_fourcc(*'mp4v') کدک مورد استفاده برای فشرده‌سازی ویدیو (در اینجا از کدک MP4 استفاده شده است.)
- Fps نرخ فریم ویدیوی خروجی که از ویدیوی ورودی گرفته شده است.
- (width, height) اندازه فریم‌های ویدیوی خروجی (۷۲۰x۱۲۸۰ پیکسل.)

```
roi_points = np.array([[300, 720], [550, 300], [720, 300], [970, 720]], np.int32)
```

این خط محدوده‌ی ROI را به‌صورت یک چندضلعی تعریف می‌کند. roi_points یک آرایه NumPy است که شامل مختصات گوشه‌های چندضلعی است. (محدوده خطر سبز رنگ تصویر زیر)

مختصات به‌صورت (x, y)هستند و نشان‌دهنده‌ی نقاطی در تصویر هستند که محدوده‌ی ROI را تشکیل می‌دهند.



```
object_width = 50
focal_length = 1000
dist_ = 12
allowed_classes = [0,14,15,16,17,18,19,20,21,22]
```

برای محاسبه فاصله ی واقعی هر شیء در تصویر عرض واقعی آن در object_width تعریف شده (در اینجا 50 سانتی متر)

که این یه عدد فرضی و مثال است (باید برای هر حیوان یک اندازه دقیق تر و مستقل در نظر گرفته شود)

همچنین برای محاسبه فاصله ی واقعی از فاصله کانونی دوربین (در اینجا 1000) استفاده میشود focal_length

این میزان بسته به دوربین مورد استفاده در هر ماشین میتواند متقاوت باشد .

Dist در واقع تعداد خطوط ترمز آبی رنگ که با برخورد هر حیوان با این خط قرمز میشود (تصویر بالا) را نشان میدهد

لیست allowed_classes کلاس های حیوانات و انسان را در مدل YOLOv5 ذخیره می کند و سیستم با توجه به این لیست تشخیص را فیلتر می کند (تشخیص فقط انسان و حیوانات انجام می شود)

0 انسان ، 14 گربه ، 15 سگ ، 16 اسب ، 17 گوسفند ، 18 گاو ، 19 فیل ، 20 خرس ، 21 زرافه ، 22پرنده

حال آماده هستیم تا پردازش ها را به طور پیوسته روی تمامی فریم های ویدیو ورودی انجام دهیم .
حلقه while بی نهایتی در نظر گرفته می شود و این حلقه تا زمانی که :

- تمامی فریم ها پردازش شوند

یا

- کاربر دکمه q را فشار دهد

ادامه پیدا میکند . در ادامه به اتفاقات و پردازش هایی که روی هر فریم صورت میگیرد خواهیم پرداخت .

```
success, frame = video.read()
    if not success:
        break
```

با دستور video.read() فریم بعدی در ویدیو خوانده می شود و مقدار آن به frame اختصاص داده میشود و success مقدار true به خود میگیرد (درصورتی که فریمی برای خواندن باقی نمانده باشد برنامه به اتمام میرسد)

```
frame = cv2.resize(frame, (1280, 720))
```

با این دستور هر فریمی که خوانده میشود به ابعاد 1280 x 720 تبدیل میشود (تا پردازش یکنواخت و سریع انجام شود)

```
cv2.polylines(frame, [roi_points], True, (0, 200, 0), 2)
```

برای رسم ناحیه ROI روی تصویر که محدوده آن را از قبل مشخص کردیم ([roi_points]) از cv2.polylines استفاده می کنیم که روی هر فریم ویدئو که چند ضلعی روی آن رسم می شود انجام می شود و مقدار True به این منظور است که محدوده آن بسته باشد . رنگ خطوط محدوده را سبز در نظر گرفته ایم و با ضخامت 2

```
  line_y1 = 720
    line_gap = 18
```

خطوط افقی ترمز که درون محدوده ROI قرار می گیرند با دستور line_y1 = 720 از پایین ترین قسمت این محدوده یا همان تصویر 720 پیکسلی شروع می شوند و فاصله هر خط از خط بعدی 18 پیکسل است

```
    line_ys = [line_y1 - i * line_gap for i in range(dist_)]
```

و در نهایت مشخص می کنیم که به ازای تعداد خطوط ترمز (dist_) هر خط 18 پیکسل بالاتر از خط قبلی می باشد

با این خطوط افقی میتوان متوجه شد حیوان در چه ناحیه ای وجود دارد و شدت خطر را تخمین زد

```
    line_colors = [(255, 0, 0) for _ in range(dist_)]
    crossed_lines = []
```

این خط برای مشخص کردن رنگ خطوط افقی است هر خط را به طور پیش فرض آبی در نظر می گیرد و در نهایت خطوطی را که حیوان از آن ها عبور کرده است را در یک لیست خالی به نام crossed_lines ذخیره می کند

```
  results = model(frame)
    detections = results.pandas().xyxy[0]
```

مدل YOLOv5 روی فریم های ویدئو اجرا می شود و و حیوانات و انسان را شناسایی می کند و برای هر یک از آنها اطلاعاتی مانند مختصات جعبه محدودکننده (محدوده شناسایی شده حیوان) ، دقت تشخیص (بین 0 و1) و کلاس اشیا (سگ ، گربه و...) را تولید می‌کند و در results ذخیره می کند و این اطلاعات را با کمک .results.pandas() به صورت یک جدول در می آوریم

```
  height, width, _ = frame.shape
    cv2.line(frame, (width // 2, 0), (width // 2, height), (160, 160, 160), 2)
```

برای تشخیص حرکت اشیا یک خط عمودی در مرکز هر فریم ویدئو میکشیم که در قسمت چپ و راست این خط و مقایسه آنها ها راحت تر حرکت اشیا رو تشخیص بدهیم . در خط اول ابعاد فریم را برمیگرداند (_ به جای channels یا همان تعداد کانال های رنگی قرار گرفته است که به آن نیاز نداریم بنابر این از _ استفاده کردیم) . cv2.line خط را با مشخص کردن نقطه شروع خط در مرکز بالای فریم ، مختصات عمودی بالای فریم ، نقطه پایان خط در مرکز پایین فریم ، مختصات عمودی پایین فریم و رنگ خاکستری و با ضخامت 2 روی تصویر رسم می کند

```
for _, detection in detections.iterrows():
        xmin = detection['xmin']
        ymin = detection['ymin']
        xmax = detection['xmax']
        ymax = detection['ymax']
        score = detection['confidence']
        class_id = int(detection['class'])
```

در این قسمت از کد با استفاده از :detections.iterrows() اطلاعات تمام اشیا را که در جدول detections بدست آوردیم (مختصات اشیا و دقت و کلاس اشیا) را پردازش می کند .

```
if class_id not in allowed_classes:
     continue
```

اگر کلاس اشیا در کلاس اشیائی که مشخص کردیم (انسان و حیوانات) نبود از تشخیص آن صرف نظر می کنیم .

```
if score >= 0.4:
      centroid_x = int((xmin + xmax) / 2)
      centroid_y = int((ymin + ymax) / 2)
```

scoreدقت تشخیص اشیا است که از جدولdetections استخراج شده که بین 0 و 1 است هرچه به 1 نزدیک تر باشد اطمینان بیشتری نصبت به وجود حیوان یا انسان پیدا می کنیم و هر چه به 0 نزدیک تر باشه احتمال وجود آن کم تر است برای دقت تشخیص آستانه ای در نظر گرفته ایم اگر بیشتر از 0.4 باشد تشخیص شی معتبر است و اگر کمتر باشد از پردازش آن صرف نظر می شود . حال مختصات مرکز جعبه محدود کننده اشیائی که دقت آنها مناسب است را بدست می آوریم برای بدست آوردن مکان دقیق اشیا در تصویر میانه افقی **xmin** و **ymin**مختصات گوشه بالا ، چپ جعبه و میانه عمودی **xmax**و **ymax**مختصات گوشه پایین ، راست جعبه را محاسبه می کنیم.

```
if cv2.pointPolygonTest(roi_points, (centroid_x, centroid_y), False) > 0:
       cv2.circle(frame, (centroid_x, centroid_y), 5, (0, 255, 0), -1)
```

با استفاده از تابع cv2.pointPolygonTest چک می کنیم که آیا مختصات نقطه مرکزی حیوان در ناحیه ROI هست یا خیر اگر مقدار بازگشتی اگر **کوچک‌تر** از 0 باشد نقطه **خارج از** ROI است ، اگر **مساوی** 0 باشد نقطه **روی مرز** ROI است و اگر **بزرگ‌تر** از 0 باشد نقطه **داخل** ROIاست و در تصویر دایره ای با شعاع 5 و رنگ سبز و ضخامت 1- (دایره تو پر) رسم می شود تا مکان آن دقیق مشخص شود .

```
for i, line_y in enumerate(line_ys):
    if ymax >= line_y:
        line_colors[i] = (0, 0, 255)
        crossed_lines.append(i + 1)
```

در این قسمت حیوان تشخیص داده شده از هر خط ترمز آبی رنگ که در محدوده ROI هست عبور کند (ymax مختصات گوشه پایین راست بزرگ تر یا مساوی مختصات خط افقی مورد نظر شود) رنگ آن خط به قرمز تبدیل میشود و سپس شماره آن به لیست **crossed_lines** اضافه می‌شود .

```
distance = (object_width * focal_length) / (xmax - xmin)
```

با استفاده از این فرمول فاصله حیوان را با توجه به اندازه آن از دوربین محاسبه می کنیم . پارامتر های آن:

- **object_width** : عرض واقعی شیء
- **focal_length** : فاصله کانونی دوربین (در اینجا برابر 1000 است)
- **xmax - xmin** : مختصات گوشه بالا چپ و پایین راست جعبه

```
    cv2.rectangle(frame, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (0, 0, 255), 2)
    cv2.putText(frame, f"Dist: {distance:.2f} cm", (int(xmin), int(ymin) - 10),
                cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 255, 255), 1)
```

با استفاده از تابع cv2.rectangle روی حیوان تشخیص داده شده با توجه به مختصات آن یک محدوده مستطیل شکل با رنگ قرمز و زخامت خطوط 2 رسم می کنیم (جعبه محدود کننده) و با استفاده از تابع cv2.putText فاصله حیوان از دوربین را با دو رقم اعشار ، 10 پیکسل بالا تر از مستطیل نمایش می دهیم سپس فونت ، اندازه(0.3) ، رنگ (زرد) و ضخامت (1) متن را هم مشخص می کنیم .

```
else:
    cv2.rectangle(frame, (int(xmin), int(ymin)), (int(xmax), int(ymax)), (255, 0, 0), 2)
```

حال اگر حیوان تشخیص داده شده داخل محدوده RIO نباشد جعبه محدود کننده آن با رنگ آبی نمایش داده می شود (بدون محاسبه فاصله) .

```
for i, line_y in enumerate(line_ys):
    line_length_reduction = i * 21
    start_x = roi_points[0][0] + line_length_reduction // 2
    end_x = roi_points[3][0] - line_length_reduction // 2
    cv2.line(frame, (start_x, line_y), (end_x, line_y), line_colors[i], 2)
```

در این قسمت میخوایم خطوط افقی در RIO را از پایین به بالا کوچک تر کنیم (مانند تصویر صفحه 20)

به ازای اندیس هر خط و مختصات آن حلقه for اجرا می شود . اندیس هر خط را در 21 پیکسل ضرب می کنیم تا هر خط 21 پیکسل کوچک تر خط پایینی خود شود و نقطه شروع و پایان خط را مشخص می کنیم

- نقطه شروع : قسمت گوشه پایین چپ محدوده ROI را با نصف مقدار کاهش طول خط جمع می کنیم (به این اندازه جلو می رویم) و از آن نقطه شروع می کنیم .
- نقطه پایان : قسمت گوشه پایین راست محدوده ROI را با نصف مقدار کاهش طول خط تفریق می کنیم (به این اندازه عقب تر می رویم) تا آن نقطه امتداد میدهیم .

به این صورت خطوط بالاتر را کوتاه تر می کنیم و در آخر با تابع cv2.line خط ها را با رنگ مشخص شده که در اینجا آبی هست و ضخامت 2 می کشیم .

```
 if crossed_lines:
    cv2.putText(frame, 'BRAKE', (1000 - 25, 100 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0,
0), 2)
```

اکنون عبور حیوانات از خطوط را بررسی می کنیم و نصبت به فاصله آن از خودرو هشدار متناسب با وضعیت را به راننده نشان می دهیم .

اگر crossed_lines (شماره خطوطی که حیوان از آنها عبور کرده است) مقدار داشته باشد این قسمت از کد اجرا می شود

پیغامی حاوی متن breake در مختصات (1000 - 25, 100 - 10) و با فونت HERSHEY_SIMPLEX ، اندازه 0.6 ، رنگ مشکی ، ضخامت 2 برای راننده نمایش داده می شود

```
crossed_lines_str = ', '.join(str(line_num) for line_num in crossed_lines)
    cv2.putText(frame, str(max(crossed_lines)), (1000, 100 + 30), cv2.FONT_HERSHEY_SIMPLEX,
0.6, (0, 255, 255), 2)
```

در ادامه خطوط عبور شده را با ' ,' از هم جدا می کند و شماره بیشترین خطی که حیوان از آن عبور کرده را با فونت مشخص ، در مختصات (1000, 100 + 30) ، اندازه 0.6 ، رنگ زرد و ضخامت 2 نمایش می دهد .

```
if max(crossed_lines) == 4 and max(crossed_lines) <= 5:
    cv2.putText(frame, 'FORWARD COLLISION WARNING', (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 2, cv2.LINE_AA)
elif max(crossed_lines) == 6 and max(crossed_lines) <= 8:
    cv2.putText(frame, 'COLLISION WARNING SEVERE', (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 2, cv2.LINE_AA)
elif max(crossed_lines) >= 9 and max(crossed_lines) <= 11:
    cv2.putText(frame, 'PAY ATTENTION & TAKE CONTROL', (10, 100), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (0, 0, 255), 2, cv2.LINE_AA)
elif max(crossed_lines) >= 11:
    cv2.putText(frame, 'EMERGENCY STOPPING ..!!', (10, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
(0, 0, 255), 2, cv2.LINE_AA)
```

با توجه به شماره خطوط عبور شده پیغام های مختلفی برای راننده ارسال می شود

اگر شماره خطوط بین 4 و5 باشد ⟵ متن 'FORWARD COLLISION WARNING' در مختصات (10, 100) ، با فونت مشخص شده ،اندازه 0.5 رنگ قرمز نمایش داده می شود . cv2.LINE_AAمتن را با لبه های صاف نمایش می دهد.

اگر شماره خطوط بین 6 و 8 باشد ⟵ متن 'COLLISION WARNING SEVERE' نمایش می دهد.

اگر شماره خطوط بین 9 و 11 باشد ⟵ متن 'PAY ATTENTION & TAKE CONTROL' نمایش می دهد.

اگر شماره خطوط بیشتر از 11 باشد ⟵ متن 'EMERGENCY STOPPING ..!!' نمایش می دهد.

```
cv2.imshow("Video", frame)
```
فریم پردازش شده در یک پنجره با اسم "Video", نمایش داده می شود

```
_output1.write(frame)
```
فریمی که پردازش شده و روی آن تغییرات انجام شده (خطوط ، متن ها و جعبه محدود کننده روی آن رسم شده) را به ویدئوی خروجی اضافه می کنیم .

```
if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```
همان طور که در اوایل توضیحات گفته شد کاربر با فشردن کلید 'q' میتواند از حلقه while خارج شود .

```
video.release()
```
زمانی که حلقه while به اتمام برسد و پردازش و عملیات ها اجرا شود ویدئوی ورودی آزاد می شود .

```
_output1.release()
```
تمام فریم ها که ذخیره شوند ویدئوی خروجی بسته می شود .

```
cv2.destroyAllWindows()
```
و در نهایت تمامی پنجره های باز را می بندیم تا فضا آزاد شود .

"در دنیایی که فناوری هر روز مرزهای جدیدی را می‌شکند، تلاش ما گامی کوچک در جهت ایمنی بیشتر و حفظ جان هاست. امید است این مطالعات بتوانند الهام بخش نوآوری های بیشتری در مسیر ساخت دنیایی امن تر برای همه باشند".