جمهورية العراق

وزارة التعليم العالي والبحث العلمي

الجامعة ديالى

كلية التربية للعلوم الصرفة

قسم علوم الحاسبات

Smart Object Detection System using Python and YOLO3 to help Blind People

البحث مقدم الى قسم علوم الحاسبات/كلية التربية للعلوم الصرفة بجامعة ديالى

وهو جزء من متطلبات نيل شهادة البكالوريوس في قسم الحاسبات

مقدم من قبل الطالب

قتيبة طه اسعد

بأشراف

م. د سعد عبد العزيز شعبان

2026

الأية

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

(يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ ۚ)

صدق الله العضيم

(المجادلة: 11)

اقرار المشرف

اشهد بان هذا المشروع بعنوان :

(نظام مطور لفهرسة الكتب والرسائل والاطاريح في كلية التربية للعلوم الصرفة)

من اعداد الطالب :

1. قتيبة طه أسعد

تحت اشراف :

م.د سعد عبد العزيز شعبان

أشهد بأن اعداد هذه المشروع قد جرى تحت اشرافي في جامعة ديالى / كلية التربية للعلوم الصرفة / قسم علوم الحاسبات وهي جزء من متطلبات نيل درجة البكالوريوس في علوم الحاسبات.

التوقيع :

الاسم :

المرتبة العلمية :

التاريخ :

الاهداء

إلى من أضاءوا لي دروب العلم والمعرفة..

إلى نبع الحنان وعنوان الأمان، إلى من سهرت الليالي لتراني في هذا المقام.. أمي الغالية .إلى من شددت به عضدي، ومن كان لي السند والقدوة في الكفاح.. أبي العزيز .إلى رفقاء الدرب وإخوتي، الذين شاركوني لحظات التعب والفرح ..الذين غرسوا فينا بذور الطموح والدعوات المستجابة ..إلى كل من آمن بقدراتي وشجعني بكلمة أو دعاء..

شكر و تقدير..

الحمد لله الذي بنعمته تتم الصالحات، والصلاة والسلام على معلم البشرية الأول محمد صلى الله عليه وسلم.. أما بعد،

فإنني أتقدم بجزيل الشكر وعظيم الامتنان إلى صرحي العلمي الشامخ **(جامعة ديالى/الكلية التربية للعلوم الصرفة)**، التي احتضنت طموحي ووفرت لي البيئة الخصبة للتعلم والبحث.

كما أخص بالشكر والتقدير أستاذي المشرف الفاضل **(م.د سعدعبد العزيز شعبان)**، الذي لم يبخل عليّ بتوجيهاته السديدة وخبرته القيمة، وكان لتوجيهاته الأثر البالغ في إخراج هذا البحث بالصورة المطلوبة.

والشكر موصول إلى جميع أساتذتي الأجلاء في قسم **(الحاسبات)** الذين نهلت من علمهم طوال سنوات الدراسة.

ولا يفوتني أن أشكر كل من ساعدني في إنجاز هذا النظام الذكي لمساعدة المكفوفين، سواء بتقديم المشورة التقنية أو المساعدة في جمع البيانات، سائلاً المولى عز وجل أن يجعل هذا العمل خالصاً لوجهه الكريم، ونافعاً لخدمة الإنسانية.

المحتويات

ملخص البحث

تعد استقلالية الحركة والتنقل من أكبر التحديات التي تواجه الأفراد ذوي الإعاقة البصرية في حياتهم اليومية، لا سيما عند صعوبة التعرف على الاشياء يهدف هذا البحث إلى تطوير نظام ذكي للكشف عن الأجسام (Smart Object Detection System) يعتمد على تقنيات الرؤية الحاسوبية (Computer Vision) والتعلم العميق (Deep Learning) لمساعدة المكفوفين في التعرف على الاشياء وتحديد مواقعها في الوقت الحقيقي.

اعتمدت الدراسة على خوارزمية YOLOv3 (You Only Look Once) نظراً لكفاءتها العالية في السرعة والدقة، مع دمج معمارية C3Ghost لتقليل التعقيد الحسابي، مما يسمح بتشغيل النظام على الأجهزة ذات الموارد المحدودة مثل الهواتف الذكية. تم استخدام لغة البرمجة Python كبيئة تطوير أساسية، بالاعتماد على مكتبة OpenCV للمعالجة المسبقة للصور ومكتبة PyTorch لبناء وتدريب الشبكة العصبية الالتفافية.(CNN)

أظهرت النتائج أن النظام المقترح قادر على كشف الاشياء وتصنيفها بدقة عالية (mAP) وبسرعة معالجة تسمح بالتفاعل اللحظي، مما يوفر للمستخدم تنبيهات صوتية دقيقة حول محيطه. يساهم هذا المشروع في سد الفجوة بين التقنيات المتقدمة واحتياجات ذوي الإعاقة البصرية، مما يعزز من أمانهم واستقلاليتهم في البيئات الحضرية المزدحمة.

الفصل الاول

1-1 مقدمة

لقد كان التعرف على الأشياء والكشف عنها مشكلة هامة لفترة طويلة للعديد من الصناعات المختلفة، بما في ذلك المجال الطبي، وصناعة الأمن، وقطاع النقل. ومع ذلك، فإن هذه المسألة تأخذ مستوى أكبر من الأهمية بالنسبة للأشخاص المكفوفين أو الذين لديهم شكل آخر من أشكال ضعف البصر.

ووفقاً لتقديرات منظمة الصحة العالمية، فإن 2.2مليار شخص على الأقل يعانون من بعض أشكال ضعف الرؤية القريبة أو البعيدة، مع خسائر في الإنتاجية السنوية مرتبطة بها تقدر بحوالي 411مليار دولار .[1]

يواجه الأشخاص المعاقون بصرياً مجموعة واسعة من الصعوبات على أساس يومي؛ ويمكن لهذه التحديات أن تتراوح من الصعوبات التي تواجه التنقل والتوجه، إلى الصعوبات التي تواجه الوصول إلى المعلومات والخدمات. ومن أهم القضايا التي تواجه الأفراد الذين يعانون من ضعف البصر على أساس يومي ما يتعلق باستخدام وسائل النقل العام، حيث تبرز صعوبة الوصول إلى وجهتهم، تليها المضايقات ومن ثم المخاوف المتعلقة بالسلامة.[2]

وفي هذا النضام يستطيع ضغيف البصر توقع الاشياء والتعرف عليها يحتاج الأشخاص الذين يعانون من ضعف البصر إلى معلومات عن محيطهم وأي تفاصيل مرئية في الاماكن العامة، مثل الجداول والمواقف، من أجل استخدام النظام بشكل مستقل وآمن. ومع ذلك، فإن غالبية الأفراد الذين يعانون من ضعف البصر يواجهون تحديات عندما يتعلق الأمر باستقلال الحافلة الصحيحة والنزول في الوجهة الصحيحة [3]. ونتيجة لذلك، قد يضطرون للاعتماد على الآخرين في التنقل، أو الحد من سفرهم، أو اختيار أنشطة أبسط لا تتطلب السفر على نطاق واسع.

وفي السنوات الأخيرة، مكن التقدم التكنولوجي من تطوير نظم للكشف عن الكائنات أكثر دقة وكفاءة؛ ولذلك كان الكشف عن الاشياء مجالاً للبحوث النشطة. وتستخدم تقنية RFIDوالنظم المدمجة على نطاق واسع للكشف عن الاشياء في نظم النقل الذكية. تستخدم هذه النظم علامات RFID المثبتة في في الكثير من الاشياء للتواصل مع أجهزة القراءة (RFID Readers) المتمركزة في اماكن معينة، مما يتيح الكشف عن الاشياء في الوقت الحقيقي وتتبع مواقعها.[4,5,6]

2.1مشكلة البحث

تتمحور مشكلة البحث في التحديات الجسيمة التي يواجهها الأشخاص ذوو الإعاقة البصرية في ممارسة حياتهم اليومية باستقلالية، ويمكن تلخيص أبعاد المشكلة في النقاط التالية

العوائق في التنقل الحر والآمن

يواجه المكفوفون صعوبة بالغة في التعرف على العوائق المحيطة بهم (مثل الحافلات، السيارات، الكراسي، أو الأشخاص) في الوقت الفعلي. الاعتماد على الوسائل التقليدية مثل "العصا البيضاء" أو "كلاب المساعدة" يظل محدوداً، حيث لا توفر هذه الوسائل تفاصيل دقيقة عن نوع العائق أو بعده، مما يزيد من احتمالية التعرض للحوادث أو الضياع

محدودية الحلول التقنية الحالية (الاعتماد على الهاردوير)

تعتمد الأنظمة السابقة بكثرة على تقنيات مثل RFID أو الحساسات فوق الصوتية(Ultrasonic) وهي أنظمة تتطلب بنية تحتية مكلفة وصيانة مستمرة، كما أنها تتأثر بالعوامل البيئية ولا توفر رؤية شاملة للمحيط كما تفعل العين البشرية

التحدي التقني (الدقة مقابل السرعة)

تكمن المشكلة التقنية في إيجاد توازن دقيق؛ فمعظم نماذج الذكاء الاصطناعي عالية الدقة تكون "ثقيلة" حسابياً وتتطلب أجهزة كمبيوتر بمواصفات عالية، بينما يحتاج الكفيف إلى نظام يعمل على "هاتفه المحمول" أو "جهاز مدمج بسيط" بسرعة فائقة (Real-time) وبدقة عالية جداً، لأن أي تأخير في الكشف قد يشكل خطراً على حياته.

3-1 اهداف البحث

يتمثل الهدف الرئيسي لهذا البحث في تطوير نظام رؤية حاسوبية ذكي ومنخفض التكلفة لمساعدة المكفوفين في التعرف على الاشياء والعوائق المحيطة في الوقت الفعلي، وينبثق عن هذا الهدف عدة أهداف فرعية:

1. الأهداف التقنية (Technical Objectives)

- تصميم بنية تحتية خفيفة الوزن (Lightweight Architecture:)بناء نموذج كشف أشياء يعتمد على تحسين خوارزمية YOLOv3 بدمج وحدات Ghost Net لتقليل التعقيد الحسابي وعدد العمليات المنطقية .(FLOPs)
- تحسين دقة وسرعة الكشف: الوصول إلى توازن مثالي بين دقة الكشف (Accuracy)وسرعة الاستنتاج (Inference Speed) لضمان تقديم تنبيهات فورية للمستخدم دون تأخير.
- تطوير نموذج :Slim Scale تعديل الهيكل الأصلي للشبكة لإنتاج نموذج نحيف يتناسب مع قدرات المعالجة المحدودة في الأجهزة المحمولة.

2. الأهداف التطبيقية (Applied Objectives)

- **دعم الحوسبة الطرفية :(Edge Computing)** تمكين النظام من العمل مباشرة على الهواتف الذكية أو الأنظمة المدمجة (Embedded Systems)دون الحاجة للاتصال الدائم بالإنترنت أو الخوادم السحابية.
- **تخصيص كشف الاشياء:** بناء قاعدة بيانات (Dataset)مخصصة لكشف الاشياءووسائل النقل العام لضمان قدرة المكفوفين على تحديد الحافلة الصحيحة وسط الزحام المروري.
- **تحسين واجهة التفاعل :(Audio Feedback)** تحويل مخرجات الكشف البصري إلى تنبيهات صوتية مفهومة تصف نوع العائق وبعده التقريبي عن المستخدم.

3. الأهداف الإنسانية والاجتماعية (Humanitarian Objectives)

- **تعزيز الاستقلالية الشخصية:** تمكين ذوي الإعاقة البصرية من التعرف على الاشياء في المحيطة وتسهيل عملية التنقل بمفردهم.
- **رفع مستوى السلامة:** تقليل مخاطر الاصطدام بالعوائق المتحركة أو الثابتة من خلال نظام إنذار مبكر يعمل بالذكاء الاصطناعي.

تحسين جودة الحياة: المساهمة في دمج المكفوفين في المجتمع عبر تسهيل وصولهم إلى الخدمات التعليمية والمهنية باستخدام تقنيات المدن الذكية.

4-1 اهمية البحث

1. الأهمية الإنسانية والاجتماعية (Social Impact & Humanitarian)

- تعزيز الاستقلالية :(Self-Reliance) يكسر المشروع حاجز الاعتماد الكلي على الآخرين في التنقل، مما يمنح الكفيف شعوراً بالحرية والثقة في ارتياد الأماكن العامة ومحطات الحافلات بمفرده.
- تحسين السلامة الشخصية: يوفر النظام "عيوناً رقمية" تنبه المستخدم للعوائق المتحركة (الاشياء) التي قد لا تدركها العصا التقليدية، مما يقلل من نسب الحوادث والإصابات.
- الدمج المجتمعي: يسهل على ذوي الإعاقة البصرية الوصول إلى أعمالهم وجامعاتهم مما يعزز من فرصهم في التعليم والعمل والمشاركة المجتمعية.

2. الأهمية التقنية والعلمية (Scientific & Technical Contribution)

- تطوير نماذج خفيفة الوزن:(Lightweight Models) تكمن الأهمية في تقديم حل لمشكلة "ثقل" خوارزميات الذكاء الاصطناعي؛ حيث يثبت البحث إمكانية دمج تقنيات مثل**GhostNet** مع **YOLOv3** للحصول على أداء عالٍ بموارد معالجة ضئيلة.

- دعم الحوسبة الطرفية :(Edge Computing) يساهم المشروع في دفع عجلة الأبحاث التي تهدف إلى تشغيل الذكاء الاصطناعي "محلياً" على الهواتف دون الحاجة لإنترنت سريع أو خوادم مكلفة، وهو توجه تقني عالمي حديث.
- **المساهمة في المدن الذكية:** يعتبر المشروع لبنة في بناء أنظمة النقل الذكية (Smart Transportation)التي تراعي احتياجات كافة فئات المجتمع.

3. الأهمية الاقتصادية (Economic Significance)

- تقليل تكلفة المساعدة: توفير بديل تقني رخيص الثمن مقارنة بتكلفة تدريب كلاب المساعدة أو توظيف مرافقين دائمين.
- زيادة الإنتاجية: وفقاً لمنظمة الصحة العالمية، يتسبب ضعف البصر في خسائر إنتاجية بمليارات الدولارات؛ لذا فإن تمكين هذه الفئة من التنقل للعمل يساهم بشكل غير مباشر في الاقتصاد.

5-1 منهجية البحث

تعتمد المنهجية المتبعة في هذا البحث على نهج تجريبي تقني يمر بخمس مراحل أساسية لضمان بناء نظام كشف دقيق وخفيف الوزن:

1. مرحلة تهيئة وإعداد البيانات Data preperation

- تحميل قاعدة بيانات مختارة (Custom Dataset)تحتوي على آلاف الصور الخاصة بالاشياء التي يستدمها الشخص العادي في يومه من زوايا وإضاءات مختلفة.
- وسم البيانات :(Data Labeling) استخدام أدوات مثل LabelImg أو Roboflow لتحديد صناديق الإحاطة (Bounding Boxes)حول الأهداف وتصنيفها.
- تعزيز البيانات :(Data Augmentation) تطبيق تقنيات مثل التدوير، تغيير السطوع، والقص العشوائي لزيادة قدرة النموذج على التعميم (Generalization)وتقليل "الأوفر فيتنج

2. بناء النموذج (Model Architecture)

- تطوير النموذج المحسن: البدء بهيكل YOLOv3 كقاعدة أساسية (Backbone).
- دمج وحدات :GhostNet استبدال الطبقات التشنجية التقليدية بوحدات **GhostConv** و **C3Ghost** لتقليل المعاملات Parameters والعمليات الحسابية.
- تحسين الهيكل: إضافة وحدة SimSPPF لزيادة كفاءة استخلاص الميزات وتطوير نسخة Slim-**scale** لضمان سرعة الكشف في الوقت الفعلي.

3. مرحلة التدريب (Model Training)

- البيئة البرمجية: استخدام لغة Python مع مكتبة CV2 مع PyTorch.
- إعدادات التدريب: تحديد المعاملات الفائقة (Hyperparameters)مثل حجم الدفعة (Size) معدل التعلم ((Learning Rate، وعدد الدورات Epoch)).
- التدريب الملحق :(Transfer Learning) البدء بأوزان مدربة مسبقاً (Pre-trained weights)على مجموعة بيانات COCO لتسريع عملية التعلم وزيادة الدقة.

4. مرحلة التقييم والاختبار (Performance Evaluation)

سيتم تقييم أداء النموذج باستخدام المقاييس الأكاديمية التالية:

- الدقة :(Precision) لقياس مدى صحة الكشف.
- الاستدعاء :(Recall) لقياس قدرة النموذج على إيجاد جميع الأهداف في الصورة.
- متوسط الدقة التقريبي **:(**mAP@0.5**)** المعيار الأساسي لتقييم كفاءة نماذج كشف الأشياء.
- التعقيد الحسابي**:** قياس عدد العمليات الحسابية (GFLOPs)وحجم النموذج بالاشتراك مع وقت الاستنتاج (Inference Time)بالملي ثانية.

الفصل الثاني
الرؤية الحاسوبية

1-2 الرؤية الحاسوبية

بأنها مجال متقاطع بين علوم الحاسوب والذكاء الاصطناعي، يهدف إلى تمكين الحواسيب والأنظمة من "رؤية" وفهم واستخراج معلومات ذات معنى من الصور الرقمية، مقاطع الفيديو،والمدخلات البصرية

بينما تحلل العين البشرية الضوء ليدرك العقل الأجسام، يقوم الحاسوب بتحليل البيانات الرقمية (البكسلات) عبر خوارزميات معقدة لمحاكاة هذه القدرة البشرية.

أهداف الرؤية الحاسوبية الرئيسية

1. الإدراك والتعرف :(Object Recognition) تحديد وتصنيف العناصر داخل المشهد (مثل تمييز حافلة عن سيارة صغيرة).
2. تحديد الموقع والكشف :(Object Detection) ليس فقط معرفة نوع الشيء، بل تحديد إحداثيات مكانه بدقة داخل الصورة.
3. تتبع الأجسام :(Object Tracking) مراقبة حركة جسم معين عبر الزمن في مقاطع الفيديو.
4. الاستعادة والتحسين :(Image Restoration) إزالة التشويش من الصور أو تحسين جودتها لاستخراج بيانات أوضح.
5. فهم المشهد :(Scene Understanding) تحليل العلاقات بين الأجسام (مثلاً: "الشخص يقف بجانب محطة الحافلة").

2-2التعلم العميق(Deep Learning)

التعلم العميق هو فرع من فروع **التعلم الآلي**(Machine Learning) ، يعتمد في بنيته على الشبكات العصبية الاصطناعية (ANN) المستوحاة من طريقة عمل الدماغ البشري.

- المفهوم :يتكون من طبقات متعددة (ومن هنا جاءت تسمية "عميق") لمعالجة البيانات. كل طبقة تستخلص ميزات معينة؛ فالطبقات الأولى قد تتعرف على "الخطوط
- والطبقات الوسطى على "الأشكال"، والطبقات الأخيرة على "الأجسام الكاملة" مثل الحافلة.

3.2 أهمية الـ CNN في معالجة الصور

1.استخراج الميزات تلقائياً :(Feature Extraction) لا تحتاج لإخبار البرنامج أن الحافلة لها نوافذ مربعة؛ الـ CNN تكتشف هذه الأنماط من تلقاء نفسها أثناء التدريب.

2.الثبات المكاني :(Spatial Invariance) تستطيع الـ CNN التعرف على الحافلة سواء كانت في يمين الصورة، يسارها، قريبة، أو بعيدة. هذا ضروري جداً لمشروعك "Smart Object Detection".

3.تقليل البيانات :(Dimensionality Reduction) عبر طبقات مثل (Pooling) تقوم الشبكة بتقليل حجم البيانات مع الحفاظ على أهم الميزات، مما يجعل المعالجة أسرع وأقل استهلاكاً للموارد.

4.الدقة العالية :تفوقت الـ CNN على البشر في مسابقات عالمية لتصنيف الصورمثل(ImageNet) ، وهي الأساس الذي بُنيت عليه خوارزميات مثل YOLOv3 التي تستخدمها

طبقات الشبكة

1.طبقة الالتفاف(Convolution Layer)

هي الطبقة الأهم والمسؤولة عن **استخراج الميزات** .(Feature Extraction)

- كيف تعمل :تمرر هذه الطبقة مصفوفات صغيرة تسمى "مرشحات (Filters) "أو (Kernels) فوق الصورة. كل مرشح يبحث عن نمط محدد.
- الوظيفة :الطبقات الأولى تكتشف الأنماط البسيطة (مثل الحواف والخطوط)، بينما تكتشف الطبقات العميقة الأنماط المعقدة (مثل شكل النافذة في الحافلة أو العجلات).
- النتيجة :تُنتج ما يسمى بـ خرائط الميزات .(Feature Maps)

2. طبقة التجميع(Pooling Layer)

تأتي عادةً بعد طبقة الالتفاف، ووظيفتها الأساسية هي التلخيص والضغط.

- **كيف تعمل :**تقوم باختيار القيمة الأكبر (Max Pooling) أو المتوسط (Average Pooling) من منطقة معينة في الصورة.
- **الأهداف:**

1. **تقليل الحجم :**تصغير أبعاد الصورة مما يسرع عملية الحساب ويقلل استهلاك الذاكرة.
2. **الثبات المكاني :**تجعل الشبكة قادرة على التعرف على الكائن حتى لو تغير مكانه قليلاً أو مال بزاوية بسيطة.
3. **منع الإفراط في التخصيص :(Overfitting)** لأنها تركز على الميزات الأهم فقط وتتجاهل التفاصيل غير الضرورية.

3.الطبقة كاملة الاتصال(Fully Connected Layer - FC)

توجد دائماً في **نهاية الشبكة**، وهي المسؤولة عن اتخاذ القرار النهائي (التصنيف).

- كيف تعمل **:**يتم تحويل مصفوفات الميزات ثلاثية الأبعاد الناتجة عن الطبقات السابقة إلى متجه واحد طويل .(Flattening) ترتبط كل خلية في هذه الطبقة بكل خلية في الطبقة التي تليها.
- الوظيفة **:**تقوم بجمع كل الميزات التي استخرجتها الطبقات السابقة (مثل: وجود عجلات + نوافذ كبيرة + شكل مستطيل) لتستنتج في النهاية" :هذه حافلة."
- النتيجة :تعطي احتمالية لكل فئة (مثل: حافلة 95%، سيارة 2%، شاحنة 3%).

4-2 كشف الأجسام (Object Detection)

هو تقنية في الرؤية الحاسوبية والذكاء الاصطناعي تهدف إلى تحديد **فئة** الكائن

(مثل: إنسان،حافلة سيارة) وتحديد **موقعه** بدقة داخل الصورة أو مقطع الفيديو.

ماهو الفرق بين• Traditional methods و• Deep learning methods؟

4-2.1الطرق التقليدية(Traditional Methods)

تعتمد هذه الطرق على ما يسمى بـ **"هندسة الميزات يدوي (Manual Feature Engineering)**

- **الآلية :**يجب على المبرمج أو الخبير تحديد الخصائص التي تميز الكائن (مثل الحواف، الألوان، أو الأشكال الهندسية) وكتابة خوارزميات لاستخراجها.
- **أمثلة :**خوارزمية SIFT، و HOG (Histogram of Oriented Gradients)، و Haar Cascades.
- نقاط الضعف * **:**صعبة جداً في البيئات المعقدة (مثل تغير الإضاءة أو زاوية التصوير).
  - تتطلب خبيراً بشرياً لتحديد الميزات لكل كائن على حدة.
  - دقتها محدودة عند التعامل مع كميات ضخمة من البيانات

4-1.2 طرق التعلم العميق(Deep Learning Methods)

تعتمد هذه الطرق على **"تعلم الميزات تلقائياً.(Automatic Feature Learning)**

- **الآلية :**بدلاً من إخبار الحاسوب "ابحث عن الخطوط الدائرية لتعرف العجلة"، نقوم بتغذية الشبكة العصبية (مثل **(CNN**) بآلاف الصور للاشياء وهي تقوم تلقائياً باكتشاف الميزات الأكثر أهمية لتمييز الحافلة.
- **أمثلة :**خوارزميات **YOLOv3**، **ResNet**، و **Faster R-CNN**.
- **نقاط القوة:**
  - دقة فائقة تتفوق أحياناً على البشر.
  - قدرة عالية على التكيف مع التغيرات في البيئة المحيطة (وهذا ضروري جداً لمشروعك في كشف الحافلات في الشارع).
  - كلما زادت البيانات، زادت دقة النموذج.

أ) R-CNN (Regions with CNN features)

- كيف تعمل **:**تقوم الخوارزمية باستخراج حوالي 2000 منطقة مقترحة (Region Proposals) من الصورة باستخدام خوارزمية تقليدية تسمى "البحث الانتقائي .(Selective Search) "ثم يتم تمرير كل منطقة بشكل منفصل عبر شبكة CNN لاستخراج الميزات وتصنيفها.
- العيوب **:**بطيئة جداً؛ لأنها تعالج 2000 منطقة لكل صورة، مما يجعلها غير صالحة للعمل في الوقت

ب Fast R-CNN

- التطوير **:**بدلاً من تمرير 2000 منطقة عبرCNN ، نقوم بتمرير **الصورة كاملة مرة واحدة** عبر CNNلاستخراج خريطة الميزات .(Feature Map) ثم نطبق الاقتراحات على هذه الخريطة مباشرة.
- المميزات **:**أسرع بكثير من R-CNN الأصلية بفضل تقنية "RoI Pooling" التي توحد أحجام المناطق المقترحة قبل تصنيفها.

ج) SSD (Single Shot MultiBox Detector

- كيف تعمل **:**تقوم بمعالجة الصورة مرة واحدة وتستخدم خرائط ميزات بأحجام مختلفة (Multi-scale) لاكتشاف الأجسام بأحجام متنوعة (صغيرة وكبيرة) في خطوة واحدة.
- المميزات **:**توازن ممتاز بين السرعة والدقة، وهي أسرع من عائلة R-CNN وتتفوق في اكتشاف الأجسام الصغيرة مقارنة بالإصدارات القديمة منYOLO

د- YOLO (You Only Look Once)

هذه هي الخوارزمية التي تستخدمها في مشروعك(YOLOv3) ، وهي الأشهر عالمياً حالياً.

- كيف تعمل :تقسم الصورة إلى شبكة .(Grid) كل خلية في الشبكة مسؤولة عن التنبؤ بالصناديق المحيطة واحتمالات الفئات في خطوة حسابية واحدة فقط.
- المميزات:
  - السرعة الهائلة :قادرة على معالجة الفيديو في الوقت الحقيقي (أكثر من 45 إطاراً في الثانية).
  - الفهم العام :تنظر إلى الصورة ككل، مما يقلل من أخطاء التنبؤ بالخلفية.

منهجية العمل(Methodology)

تعتبر منهجية العمل (Methodology) هي الخريطة التقنية التي توضح كيف ينتقل النظام من مجرد "صورة" تلتقطها الكاميرا إلى "تنبيه صوتي" يساعد الكفيف. في مشروعك، تتبع المنهجية مساراً منطقياً يتكون من خمس مراحل أساسية:

المرحلة الأولى: جمع وإعداد البيانات(Data Acquisition & Preparation)

هذه المرحلة هي حجر الأساس لتدريب نموذج:YOLOv3

- **جمع البيانات :**تم جمع قاعدة بيانات تحتوي على صور متنوعة للاشياء في ظروف مختلفة (نهار، ليل، زوايا تصوير مختلفة).
- **توسيم البيانات :(Labeling)** تم استخدام أدوات مثل LabelImg أو Roboflow لرسم صناديق محيطة (Bounding Boxes) حول االاشياء وتحديد فئتها.
- **تقسيم البيانات :**تم تقسيم البيانات إلى ثلاثة أقسام:
  1. **تدريب :(Training)** لتعليم النموذج.
  2. **تحقق :(Validation)** لضبط إعدادات النموذج أثناء التدريب.
  3. **اختبار :(Testing)** لتقييم الدقة النهائية.

المرحلة الثانية: المعالجة المسبقة (Preprocessing)

قبل إرسال الصورة للنموذج، يتم تجهيزها لضمان أفضل أداء:

- **توحيد الحجم :(Resizing)** تحويل كل الصور إلى مقاس ثابت (مثل 640×640بكسل)
- **التطبيع :(Normalization)** تحويل قيم البكسلات إلى نطاق بين 0و 1لتسريع الحسابات الرياضية في.PyTorch
- **تعزيز البيانات :(Augmentation)** توليد صور جديدة عبر التدوير وتغيير السطوع لزيادة قدرة النموذج على التعرف في الظروف الصعبة.

المرحلة الثالثة: هيكلية النظام (System Architecture) باستخدام YOLOv3 & C3Ghost

هنا يتم بناء "العقل" الذي سيقوم بالكشف:

- **Backbone العمود الفقري :** استخدام نموذج YOLOv3 المدمج مع وحدات **C3Ghost** لاستخراج الميزات (Features) بفعالية عالية وبأقل عدد من العمليات الحسابية.
- **Neck الرقبة :** لدمج الميزات المستخرجة بأحجام مختلفة لاكتشاف الحافلات القريبة والبعيدة.
- **Head الرأس :** المسؤول عن التنبؤ النهائي بمواقع الصناديق ونوع الكائن

المرحلة الرابعة: الاختبار والاستنتاج(Inference Phase)

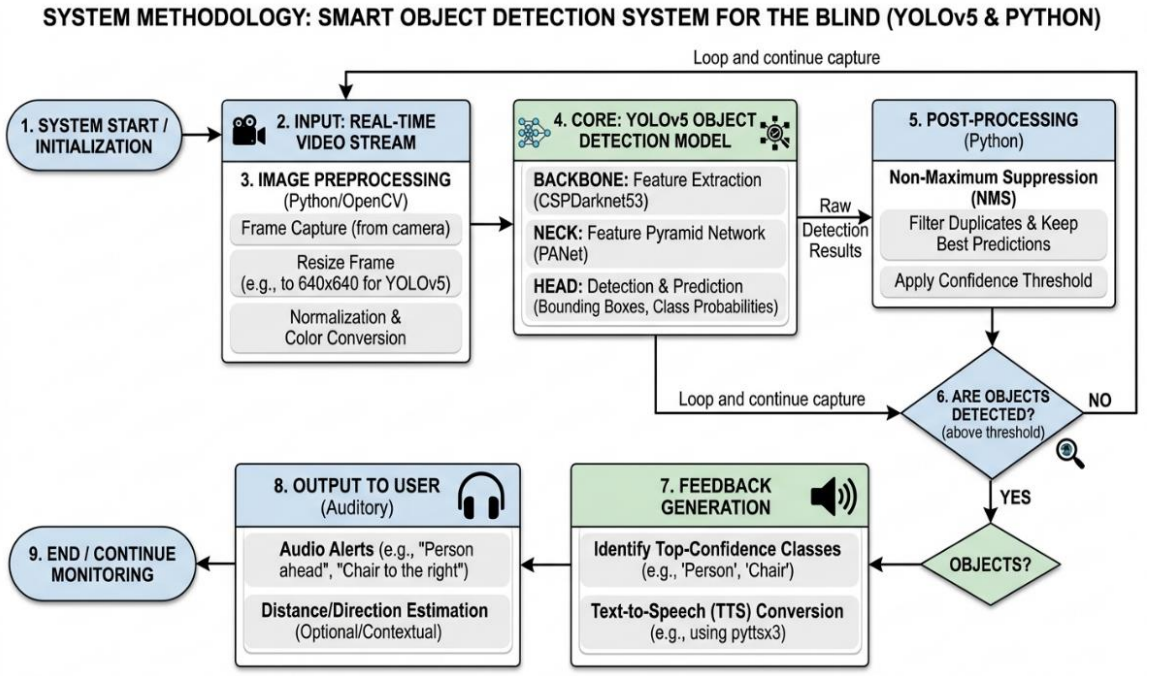
عند تشغيل النظام بشكل فعلي:(Real-time)

1. **التقاط الفيديو :** يتم جلب الإطارات (Frames) من كاميرا الجهاز باستخدام.OpenCV
2. **التنبؤ :(Prediction)** يقوم النموذج المحمل عبر PyTorch بتحديد مكان الحافلة ورسم صندوق محيط بها.
3. **فلترة النتائج :(NMS)** استخدام تقنية (Non-Maximum Suppression) لاختيار أفضل صندوق محيط وحذف الصناديق المتكررة لنفس الحافلة.

المرحلة الخامسة: واجهة التفاعل الصوتي(Audio Feedback)

هذه المرحلة هي التي تربط التقنية بالمستخدم الكفيف:

- يتم تحويل نتائج الكشف كلمة ("Bus")إلى ملف صوتي أو تنبيه صوتي فوري يخبر المستخدم: "حافلة تقترب."
- يتم دمج هذا النظام ليعمل بشكل متزامن مع معالجة الفيديو لضمان عدم وجود تأخير.(Latency)



الفصل الثالث

لغة بايثون

مقدمة

تعتمد المرحلة العملية في هذا المشروع على بناء بيئة برمجية متكاملة تدمج بين قوة الحسابات الرياضية وسرعة المعالجة البصرية، حيث تبدأ الرحلة من إعداد قاعدة البيانات وتوسيمها بدقة لتكون صالحة لتدريب الأوزان داخل إطار عمل **python** يتم التركيز في الجانب التطبيقي على توظيف خوارزمية **YOLOv3** كالمحرك الأساسي لعمليات الكشف اللحظي، مع إدخال تعديلات جوهرية على بنيتها الهيكلية باستخدام وحدات **C3Ghost** لضمان الحصول على نموذج ذكي يتميز بخفة الوزن والقدرة على العمل فوق الأجهزة ذات الإمكانيات المحدودة دون استهلاك مفرط للطاقة أو الذاكرة.

وتلعب مكتبة **OpenCV** دور الجسر الرابط في هذا الجانب العملي، إذ تتولى مسؤولية إدارة تدفق الفيديو الملتقط من الكاميرا ومعالجته لحظياً عبر سلسلة من العمليات الرياضية مثل تغيير الأبعاد وتطبيع قيم البكسلات لتتوافق مع مدخلات الشبكة العصبية. وبمجرد أن يستقبل النموذج هذه البيانات، يقوم بإجراء عمليات التنبؤ ورسم الصناديق المحيطة بالأجسام المكتشفة، ثم يتم تحويل هذه النتائج الرقمية إلى أوامر برمجية تستدعي وظائف التنبيه الصوتي، مما يحول مخرجات الخوارزمية المعقدة إلى وسيلة مساعدة ملموسة تمكن الكفيف من إدراك محيطه والتفاعل مع وسائل النقل بكفاءة عالية وأمان تام.

python 1-3

لغة بايثون

بايثون هي لغة برمجة **عالية المستوى(High-level)** ، **مفسرة(Interpreted)** ، و**متعددة الأغراض** . ابتكرها المبرمج الهولندي "جيدو فان روسوم (Guido van Rossum) "عام 1991، وصممها ليكون شعارها الأول هو **"القراءة قبل الكتابة"**، أي أن الكود البرمجي فيها يجب أن يكون سهلاً وواضحاً كاللغة الإنجليزية.

أهم مميزات بايثون

- **سهولة التعلم والقراءة :**بساطة قواعدها (Syntax) تجعلها الخيار الأول للمبتدئين وللباحثين في المجالات العلمية.
- **لغة مفتوحة المصدر :**بايثون مجانية تماماً، ومدعومة بمجتمع ضخم من المطورين الذين يساهمون في تحسينها باستمرار.
- **المكتبات الضخمة :**تمتلك بايثون آلاف المكتبات الجاهزة التي توفر عليك كتابة الأكواد من الصفر، مثل:
  - (PyTorch) (OpenCV) التي تستخدمها في رؤية الحاسوب.
  - (NumPy) (Pandas) لتحليل البيانات.
- **تعمل على كل المنصات :**يمكنك كتابة الكود على Windows وتشغيله على Linux أو Mac أو حتى أنظمة الأجهزة المدمجة.(Embedded Systems)

إليك الأسباب الرئيسية التي جعلت بايثون الخيار الأول لمكتبات مثل OpenCVو PyTorch و NumPy وMatplotlib

1.التكامل مع لغتي (C) و (C++)الأداء العالي

رغم أن بايثون لغة "بطيئة" نسبيًا مقارنة باللغات منخفضة المستوى، إلا أن المكتبات مثل **OpenCV** و **PyTorch**و **NumPy**مكتوبة في جوهرها بلغة **C++**أو **C**لضمان السرعة القصوى في العمليات الحسابية.

- **السبب :**توفر بايثون واجهة برمجية (Wrapper) تسمح للمبرمج بكتابة كود بسيط وسهل، بينما تُنفذ العمليات المعقدة (مثل معالجة ملايين البكسلات أو ضرب المصفوفات الضخمة) في الخلفية بسرعة لغة

2.التعامل مع البيانات كمصفوفات(NumPy)

تعتبر مكتبة **NumPy**هي القاعدة الأساسية التي تُبنى عليها بقية المكتبات.

- **السبب :**الصور في **OpenCV**تُعامل كمصفوفات أرقام، والأوزان في **PyTorch**هي مصفوفات (Tensors).بايثون تجعل التعامل مع هذه المصفوفات رياضيًا أمرًا غاية في السهولة والبديهية مقارنة بلغات مثل Java أو.C++

3.الإنتاجية وسرعة التطوير(Productivity)

في مشاريع مثل مشروعك (كشف الاشياء للمكفوفين)، يهمك تجربة الخوارزميات وتعديلها بسرعة.

- **السبب :**بايثون تتيح لك كتابة في 5 أسطر ما قد يتطلب 50 سطرًا في لغة أخرى. هذا يسمح للباحثين والطلاب بالتركيز على **المنطق والذكاء الاصطناعي** بدلًا من الانشغال بتعقيدات الذاكرة وإدارة النظام.

4.النظام البيئي المتكامل(Ecosystem)

هذه المكتبات تعمل معًا وكأنها قطعة واحدة:

- **OpenCV:**تلتقط الصورة من الكاميرا وتحولها لمصفوفة.
- **NumPy:**تعالج هذه المصفوفة وتجري عليها عمليات رياضية.
- **PyTorch:**تأخذ المصفوفة وتمررها عبر الشبكة العصبية (YOLO) لكشف الأجسام.
- **Matplotlib:**تعرض النتائج والرسوم البيانية لمراقبة دقة النموذج.
- **السبب :**هذا التكامل "السلس" لا يتوفر بنفس القوة في أي لغة برمجة أخرى.

5.دعم المجتمع والمكتبات الجاهزة

- **السبب :**نظرًا لأن أغلب أبحاث الذكاء الاصطناعي تُنشر بكود بايثون، فإن أي مشكلة تواجهك في **PyTorch**أو **OpenCV**ستجد لها آلاف الحلول الجاهزة على الإنترنت، وهذا عامل حاسم في نجاح المشاريع التقنية.

3- openCV 2

تعتبر **OpenCV** الأداة الأقوى في مرحلة **المعالجة المسبقة(Preprocessing)** ، وهي الخطوة التي تسبق إدخال البيانات إلى نموذج الذكاء الاصطناعي مثل .(YOLOv3) تهدف هذه المرحلة إلى تنظيف الصورة وتوحيد مقاييسها لرفع دقة الكشف.

كيف تتعامل المكتبة مع الصور والفيديو في هذه المرحلة:

1.التعامل مع الصور(Image Preprocessing)

تتضمن المعالجة المسبقة للصور عدة عمليات أساسية تقوم بها:OpenCV

- **تغيير الحجم:(Resizing)**

تحتاج نماذج التعلم العميق إلى صور ذات أبعاد ثابتة مثل .(640×640) تستخدم OpenCVدالة ()cv2.resizeلضمان توافق الصورة مع مدخلات الشبكة العصبية.

- **تحويل مساحات الألوان:(Color Space Conversion)**

أحياناً يكون من الأسهل كشف الأجسام في التدرج الرمادي (Gray Scale) أو في مساحة ألوان **HSV**للكشف بناءً على اللون. يتم ذلك عبر .()cv2.cvtColor

- **التطبيع:(Normalization)**

تحويل قيم البكسلات من نطاق 0 إلى (255إلى نطاق 0) إلى (1أو $1-$) إلى (1) لمساعدة النموذج على التعلم بشكل أسرع واستقرار الحسابات الرياضية.

- **إزالة الضجيج:(Denoising)**

استخدام فلاتر مثل (Gaussian Blur) أو (Median Blur) لتنعيم الصورة وتقليل "الشوشرة" التي قد تربك النموذج.

2.التعامل مع الفيديو(Video Preprocessing)

الفيديو بالنسبة لـ OpenCV هو مجرد **سلسلة من الصور المتتابعة** .(Frames)تتعامل معه المكتبة كالتالي:

- **استخراج الإطارات:(Frame Extraction)**

باستخدام cv2.VideoCapture()، تقوم المكتبة بقراءة الفيديو إطاراً بإطار (مثلاً 30 إطاراً في الثانية). كل إطار يُعامل كصورة مستقلة تُطبق عليها عمليات المعالجة المذكورة أعلاه.

- **تقليل معدل الإطارات:(FPS Reduction)**

في مشروعك للمكفوفين، قد لا نحتاج لمعالجة كل الإطارات لتوفير الطاقة والجهد الحسابي؛ لذا يمكن لـ OpenCVتخطي بعض الإطارات ومعالجة إطار واحد كل 3 إطارات مثلاً.

- **طرح الخلفية:(Background Subtraction)**

تقنية تستخدم لكشف الأجسام المتحركة فقط عبر مقارنة الإطار الحالي بالإطار السابق، مما يساعد في عزل الأجسام المتحركة (مثل شيء يقترب) عن العناصر الثابتة (مثل الأشجار).

3.تقنيات تعزيز البيانات(Data Augmentation)

أثناء تدريب نموذجك، تُستخدم OpenCV لإنشاء نسخ "مشوهة" من الصور الأصلية لزيادة قدرة النموذج على التعرف:

- **التدوير (Rotation) والقلب.(Flipping)**
- **تغيير السطوع والتباين.(Brightness & Contrast)**
- **القص العشوائي.(Random Cropping)**

pyTorch 3-3

تعتبر **PyTorch**هي البيئة الحاضنة (Framework) التي بُني عليها نموذج **YOLOv3**. هي المسؤولة عن إدارة الحسابات الرياضية المعقدة (المصفوفات) وتوزيعها على المعالج الرسومي (GPU) لضمان السرعة.

مبدأ عمل PyTorch مع YOLOv3

تتعامل PyTorch مع البيانات كـ **Tensors**(مصفوفات متعددة الأبعاد). عند تشغيلYOLOv3 ، تقوم PyTorchبالآتي:

1. **بناء الهيكل :(Architecture)** تحويل ملفات الإعدادات مثل (yolov3s.yaml)إلى طبقات برمجية (Convolution, Pooling, C3).
2. **التنفيذ الديناميكي :(Dynamic Computation Graph)** تسمح PyTorch بتغيير مسار البيانات أثناء التشغيل، مما يجعلها مرنة جداً في معالجة الصور ذات الأحجام المختلفة.
3. **التسريع الحسابي :**ربط الأكواد بمكتبات **CUDA**من NVIDIA لتشغيل آلاف العمليات الحسابية في وقت واحد على الـGPU ، وهو ما يمنح YOLO سرعته الفائقة

مرحلة التدريب(Training Phase)

التدريب هو عملية تعليم النموذج التعرف على "الحافلات" من خلال ملايين الأمثلة. تمر هذه العملية بالخطوات التالية:

1. **التغذية الأمامية :(Forward Pass)** تدخل الصورة إلى الشبكة، وتتوقع الشبكة مكان الحافلة ونوعها. في البداية، تكون التوقعات عشوائية تماماً.
2. **حساب الخسارة :(Loss Function)** تقارن PyTorch بين "توقع الشبكة" وبين "الحقيقة) "التي وضعتها أنت في وسم البيانات .(Label)الفرق بينهما يسمى .**Loss**
3. **الانتشار العكسي :(Backpropagation)** هنا تظهر قوةPyTorch ؛ حيث تقوم بحساب كيفية تعديل "الأوزان" داخل الشبكة لتقليل قيمة الـ Loss في المرة القادمة.
4. **المحسن :(Optimizer)** تستخدم خوارزميات مثل (SGD) أو (Adam) لتحديث الأوزان فعلياً. تكرر هذه العملية لعدة دورات (**Epochs**) حتى تصبح الدقة عالية.

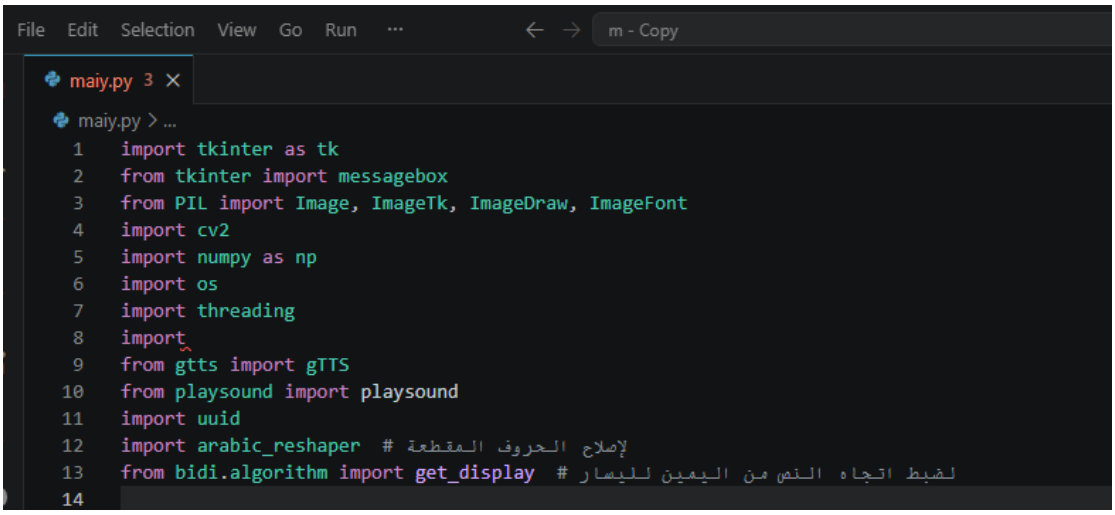
الفصل الرابع

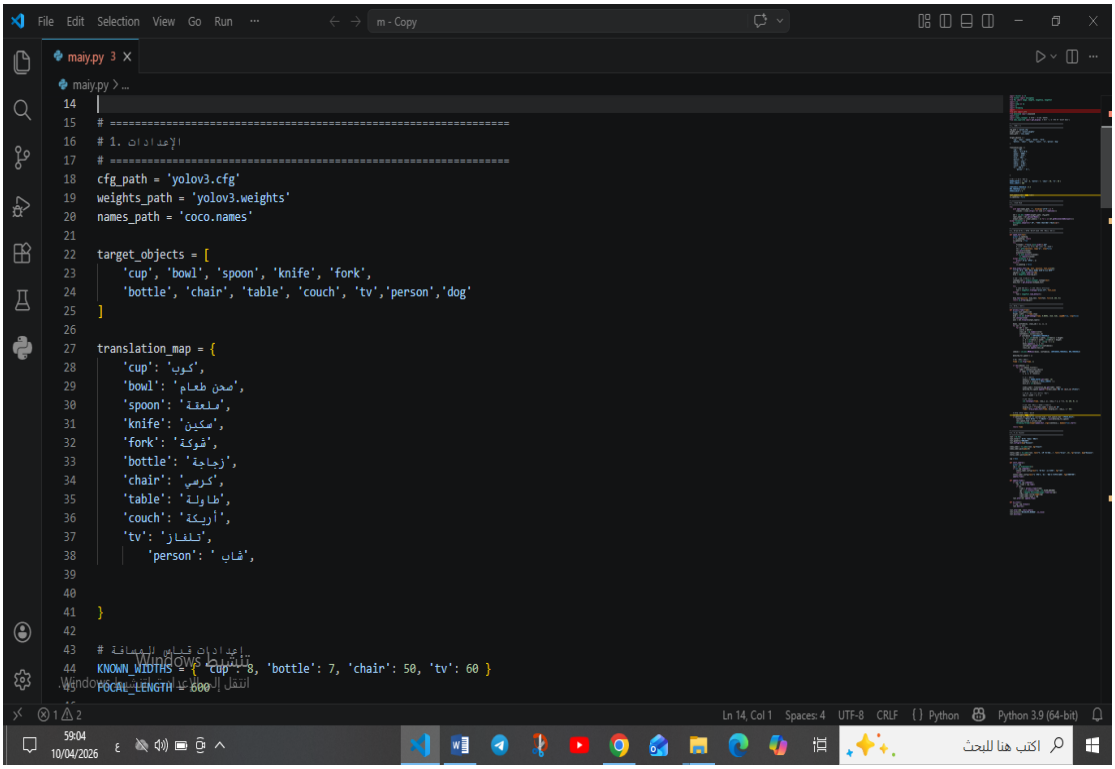
الجانب العملي

مقدمة

تعتمد المرحلة العملية في هذا المشروع على بناء بيئة برمجية متكاملة تدمج بين قوة الحسابات الرياضية وسرعة المعالجة البصرية، حيث تبدأ الرحلة من إعداد قاعدة البيانات وتوسيمها بدقة لتكون صالحة لتدريب الأوزان داخل إطار عمل **python** يتم التركيز في الجانب التطبيقي على توظيف خوارزمية **YOLOv3** كالمحرك الأساسي لعمليات الكشف اللحظي، مع إدخال تعديلات جوهرية على بنيتها الهيكلية باستخدام وحدات **C3Ghost** لضمان الحصول على نموذج ذكي يتميز بخفة الوزن والقدرة على العمل فوق الأجهزة ذات الإمكانيات المحدودة دون استهلاك مفرط للطاقة أو الذاكرة.ساعدة ملموسة تمكن الكفيف من إدراك محيطه والتفاعل مع وسائل النقل بكفاءة عالية وأمان تام.

المكتبات





1. مسارات ملفات النموذج (Model Files)

الأسطر من 18 إلى 20 تحدد الملفات الأساسية التي يحتاجها محرك YOLO3 ليعمل:

- 'cfg_path = 'yolov3.cfg: هذا ملف الإعدادات، ويحتوي على بنية الشبكة العصبية (عدد الطبقات، سرعة التعلم، وحجم الصور المدخلة).
- 'weights_path = 'yolov3.weights: هذا هو "عقل" النموذج، ويحتوي على الأوزان التي تم تعلمها مسبقاً من تدريب النموذج على ملايين الصور.
- 'names_path = 'coco.names: ملف نصي يحتوي على قائمة بأسماء الأشياء الـ 80 التي يستطيع نموذج YOLO الأساسي التعرف عليها (مثل: انسان، سيارة، قطة).

2.تحديد الأهداف(Target Objects)

الأسطر من 22 إلى 25 تقوم بتصفية النتائج:

target_objects: هذه مصفوفة تحتوي على قائمة بالأشياء التي تهمك فقط. بالرغم من أن النموذج يعرف 80 نوعاً، إلا أن الكود سيتجاهل أي شيء غير موجود في هذه القائمة (مثل: الكوب، الملعقة، الكرسي، الكلب، إلخ)

3.خريطة الترجمة(Translation Map)

الأسطر من 27 إلى 41 تهدف إلى تحويل المخرجات للغة العربية:

- **translation_map**: قاموس (Dictionary) يربط الكلمة بالإنجليزية (التي يخرجها النموذج) بمرادفها بالعربية.

```
42
43  # إعدادات قياس المسافة
44  KNOWN_WIDTHS = { 'cup': 8, 'bottle': 7, 'chair': 50, 'tv': 60 }
45  FOCAL_LENGTH = 600
46
47  CONFIDENCE_THRESHOLD = 0.5
48  NMS_THRESHOLD = 0.4
49  SPEECH_DELAY = 5
50
51  last_speech_time = time.time()
52  is_speaking = False
53
```

1.إعدادات قياس المسافة(Distance Estimation)

الأسطر 44 و 45 مخصصة للجانب الفيزيائي في الكود:

- **KNOWN_WIDTHS**: هذا قاموس يحتوي على العرض الحقيقي (بالسنتيمتر غالباً) للأجسام في الواقع. الكود يحتاج هذه القيم لأنه سيقارن "العرض الحقيقي" بـ "العرض بالبكسل" الذي تراه الكاميرا.
  - مثلاً: الكود يفترض أن عرض الكوب **8سم** وعرض الكرسي **50سم**.
- **FOCAL_LENGTH = 600**: هو "البعد البؤري" لعدسة كاميرتك. هذا الرقم هو ثابت رياضي يُستخدم في معادلة التناسب لحساب بُعد الجسم عن الكاميرا.

2.معايير الدقة والجودة(Detection Thresholds)

الأسطر 47 و 48 هي "الفلاتر" التي تضمن عدم ظهور نتائج خاطئة:

- **CONFIDENCE_THRESHOLD = 0.5**: حد الثقة. يخبر الكود: "لا تعترف بوجود جسم إلا إذا كنت متأكداً بنسبة **50%** أو أكثر". إذا كانت الثقة أقل، سيتم تجاهل الجسم تماماً.
- **NMS_THRESHOLD = 0.4**: اختصار لـ (Non-Maximum Suppression).
  - **الوظيفة :** عندما يتعرف الذكاء الاصطناعي على جسم (مثل "شخص")، قد يرسم حوله 5 مربعات متداخلة لنفس الشخص. هذا الفلتر يقوم بدمج هذه المربعات في مربع واحد فقط. الرقم **0.4** يحدد مدى السماح بالتداخل قبل الحذف.

3.التحكم في النطق الصوتي(Speech Control)

بما أن مشروعك يحتوي على خاصية تحويل النص إلى صوت(Text-to-Speech) ، فإن الأسطر من 49 إلى 52 تنظم هذه العملية:

- **SPEECH_DELAY = 5**: هذا هو "وقت التأخير". يخبر البرنامج ألا ينطق اسم الجسم المكتشف مرة أخرى إلا بعد مرور **5ثوانٍ**. بدون هذا السطر، سيظل البرنامج يكرر "كوب.. كوب.. كوب" بسرعة مزعجة جداً كلما رأى الكوب.
- **last_speech_time = time.time()**: يقوم بتخزين الوقت الحالي لحظة تشغيل البرنامج. سيستخدمه لاحقاً للمقارنة مع الـ SPEECH_DELAY ليعرف هل حان وقت النطق أم لا.
- **is_speaking = False**: هذا متغير منطقي .(Flag) وظيفته منع تداخل الأصوات؛ فإذا كان البرنامج ينطق جملة الآن، تكون القيمة True ليمنع نطق جملة أخرى فوقها، وعندما ينتهي يعود لـ False.

```
55  # 2. تحميل YOLO
56  # ============================================================
57  try:
58      with open(names_path, 'r', encoding='utf-8') as f:
59          classes = [line.strip() for line in f.readlines()]
60
61      net = cv2.dnn.readNet(weights_path, cfg_path)
62      layer_names = net.getLayerNames()
63      output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
64  except Exception as e:
65      messagebox.showerror("خطأ", f"فشل تحميل ملفات YOLO\n{e}")
66      exit()
67
```

1.استخراج أسماء الأشياء(Classes)

السطر 58 و 59:

- **الوظيفة :**يفتح الملف الذي يحتوي على أسماء الأشياء مثل .(coco.names)
- **التفصيل :**يقوم بقراءة كل سطر في الملف، وبواسطة (line.strip)يحذف أي مسافات زائدة أو فراغات، ثم يضعها جميعاً في قائمة (List) تسمى .classes
- **الفائدة :**هذا يتيح للكود معرفة أن الرقم "0" الذي يخرجه النموذج يعني"person" ، والرقم "1" يعني "bicycle"، وهكذا.

2.بناء الشبكة العصبية(Loading the Model)

السطر 61:

- **الوظيفة :**تحميل نموذج **YOLO**في الذاكرة باستخدام مكتبة **OpenCV**.
- **التفصيل :**تأخذ الدالة read Netملف الأوزان (.weights) الذي يحتوي على "الخبرة المكتسبة" وملف الإعدادات (.cfg) الذي يحتوي على "الهيكل الهندسي" للشبكة، وتقوم بدمجهما لإنشاء كائن (Object)يسمى .netهذا الكائن هو "العقل الذكي" الذي سيمرر الصور من خلاله لاحقاً.

3.تحديد طبقات المخرجات(Output Layers)

السطر 62 و 63:

- **الوظيفة :**تحديد "نهايات" الشبكة العصبية التي تعطي النتائج.
- **التفصيل :**شبكة YOLO تتكون من مئات الطبقات الداخلية، لكننا لا نحتاج لنتائج الطبقات الوسطى. نحن نحتاج فقط للطبقات النهائية (Unconnected Output Layers) التي تخبرنا بمواقع الأجسام.
- **الفائدة :**تحديد هذه الطبقات مسبقاً يسرع عملية المعالجة، لأننا نطلب من الشبكة أن تعطينا مخرجات هذه

4.نظام الحماية(Error Handling)

السطر 64 إلى 66:

- **الوظيفة :**التعامل مع الأخطاء المفاجئة.
- **التفصيل :**(إذا حدثت أي مشكلة في الأسطر السابقة) مثل: ملف مفقود، أو إصدار OpenCV غير متوافق، سينتقل الكود فوراً إلى هنا.

```python
# ============================================================
# 3. وظائف مساعدة (النطق المحسن ورسم النص العربي الصحيح)
# ============================================================
def speak_text(text):
    global is_speaking
    if is_speaking: return
    is_speaking = True
    try:
        filename = f"voice_{uuid.uuid4()}.mp3"
        # تحسين النطق عبر إضافة وقف بسيطة بين الكلمات
        tts = gTTS(text=text, lang='ar', slow=False)
        tts.save(filename)
        playsound(filename)
        if os.path.exists(filename):
            os.remove(filename)
    except Exception as e:
        print("خطأ في النطق:", e)
    finally:
        is_speaking = False

def draw_arabic_text(img, text, position, font_size=28):
    """دالة تعالج النص العربي ليظهر متصلاً وغير مقطع"""
    img_pil = Image.fromarray(img)
    draw = ImageDraw.Draw(img_pil)

    # إصلاح شكل الحروف واتجاهها
    reshaped_text = arabic_reshaper.reshape(text)
    bidi_text = get_display(reshaped_text)

    try:
        # يفضل وضع مسار خط يدعم العربية مثل Arial
        font = ImageFont.truetype("arial.ttf", font_size)
    except:
        font = ImageFont.load_default()

    draw.text(position, bidi_text, font=font, fill=(0, 255, 0))
    return np.array(img_pil)
```

1. دالة النطق الصوتي speak_text(text)

هذه الدالة وظيفتها أخذ الكلمة (مثل "كوب") وتحويلها إلى صوت مسموع.

- **نظام الحماية من التداخل (الأسطر 72-74):**
  - يستخدم المتغير global is_speaking. إذا كان البرنامج يتحدث بالفعل، فإن الأمر return يجعله يتجاهل أي طلب نطق جديد حتى ينتهي من الأول. هذا يمنع تداخل الأصوات (تجنب حدوث "فوضى صوتية").
- **إنشاء ملف صوتي فريد (السطر 76):**
  - filename = f"voice_{uuid.uuid4()}.mp3": يستخدم مكتبة uuid لإنشاء اسم عشوائي وفريد لكل ملف صوتي. هذا يضمن عدم حدوث تصادم إذا تم إنشاء ملفين في وقت متقارب جداً.
- **تحويل النص لصوت (الأسطر 78-79):**
  - يستخدم مكتبة gTTS (Google Text-to-Speech) مع ضبط اللغة على العربية lang='ar'. يتم حفظ النتيجة كملف MP3 مؤقت.
- **التشغيل والتنظيف (الأسطر 80-82):**
  - Playsound (filename): يقوم بتشغيل الملف ليسمعه المستخدم.
  - os.remove (filename): بمجرد انتهاء الصوت، يقوم البرنامج بحذف الملف فوراً لتوفير مساحة الذاكرة وعدم تراكم ملفات لا داعي لها.
- **الخاتمة (finallyالسطر 86) :**
  - سواء نجح النطق أو حدث خطأ، سيتم تحويل is_speaking إلى False مرة أخرى، ليعلن البرنامج أنه جاهز للنطق مرة ثانية.

2. دالة رسم النص العربي draw_arabic_text

هذه الدالة تحل مشكلة تقنية مشهورة في مكتبةOpenCV ، وهي أنها لا تدعم الأحرف العربية (تظهرها مقلوبة أو متقطعة).

- **تحويل التنسيق (السطر 90):**
  - Image.fromarray(img): مكتبة OpenCV تتعامل مع الصور كمصفوفات أرقام. هنا نقوم بتحويل الصورة إلى تنسيق مكتبة **PIL (Pillow)** لأنها بارعة في التعامل مع الخطوط والنصوص المعقدة.
- **تجهيز الرسم (السطر 91):**
  - ImageDraw.Draw(img_pil): تهيئة "لوحة رسم" فوق الصورة لنتمكن من كتابة النص عليها.

```
106  # ============================================================
107  # 4. معالجة الإطار
108  # ============================================================
109  def process_frame(frame):
110      global last_speech_time
111      height, width, _ = frame.shape
112      blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), swapRB=True, crop=False)
113      net.setInput(blob)
114      outs = net.forward(output_layers)
115
116      boxes, confidences, class_ids = [], [], []
117      for out in outs:
118          for det in out:
119              scores = det[5:]
120              class_id = np.argmax(scores)
121              confidence = scores[class_id]
122              if confidence > CONFIDENCE_THRESHOLD:
123                  cx, cy = int(det[0] * width), int(det[1] * height)
124                  w, h = int(det[2] * width), int(det[3] * height)
125                  x, y = int(cx - w / 2), int(cy - h / 2)
126                  boxes.append([x, y, w, h])
127                  confidences.append(float(confidence))
128                  class_ids.append(class_id)
129
130      indexes = cv2.dnn.NMSBoxes(boxes, confidences, CONFIDENCE_THRESHOLD, NMS_THRESHOLD)
131
132      detected_for_speech = []
133
134      # قلب الصورة للمرآة
```

1.تجهيز الصورة للذكاء الاصطناعي(Preprocessing)

الذكاء الاصطناعي لا يفهم الصور العادية مباشرة، بل يحتاج لتحويلها إلى تنسيق رياضي يسمى **Blob**.

- **السطر 111 :**يستخرج أبعاد الصورة (الطول والعرض) لاستخدامها لاحقاً في رسم المربعات.
- **السطر 112** cv2.dnn.blobFromImage **:**تقوم بثلاث عمليات حيوية:

1. **تصغير الحجم :**تحول الصورة إلى (416×416بكسل) وهو الحجم الذي يتوقعه نموذج.(YOLOv3)
2. **المعايرة :(Scaling)** تضرب قيم الألوان في) 0.00392أي تقسمها على 255) لتصبح القيم بين 0 و 1.
3. **تبديل الألوان :**تحول ترتيب الألوان من BGR إلى.RGB

2.عملية "التمرير الأمام (Forward Pass)

الأسطر 113 - 114:

هنا تبدأ عملية الحساب الفعلي داخل الشبكة العصبية:

- net.setInput(blob): نضع الصورة المجهزة داخل الشبكة.
- net.forward(output_layers): نطلب من الشبكة معالجة الصورة وإعطاء النتائج من طبقات المخرجات التي حددناها سابقاً. النتيجة outs تحتوي على آلاف التوقعات الاحتمالية.

3.تحليل النتائج وفلترتها(Decoding Outputs)

بما أن النموذج يتوقع وجود أجسام في كل مكان، نحتاج لفلترة هذه التوقعات (الأسطر 117 - 128):

- **تحديد النوع :** يبحث الكود عن أعلى درجة ثقة في مصفوفة النتائج np.argmax(scores) ليعرف هل هذا الجسم شخص أم سيارة أم كوب.
- **التحقق من العتبة (Threshold):** if confidence > CONFIDENCE_THRESHOLD: إذا كانت نسبة التأكد أكبر من 50% (القيمة التي حددناها في البداية)، يبدأ الكود في استخراج إحداثيات الجسم.
- **حساب الإحداثيات :** يقوم الكود بتحويل الإحداثيات النسبية (التي تكون بين 0 و 1) إلى إحداثيات بكسل حقيقية تناسب حجم شاشتك.(width, height)

```
137     if len(indexes) > 0:
138         for i in indexes.flatten():
139             label = classes[class_ids[i]]
140             if label in target_objects:
141                 x, y, w, h = boxes[i]
142 
143                 # حساب المسافة
144                 known_w = KNOWN_WIDTHS.get(label, 15)
145                 distance = (known_w * FOCAL_LENGTH) / w
146                 dist_cm = int(distance)
147 
148                 arabic_name = translation_map.get(label, label)
149                 detected_for_speech.append(f"{arabic_name} على بعد {dist_cm} سنتيمتر")
150 
151                 # تعديل الإحداثيات لتناسب المرآة
152                 new_x = width - (x + w)
153 
154                 # رسم المربع
155                 cv2.rectangle(frame, (new_x, y), (new_x + w, y + h), (0, 255, 0), 2)
156 
157                 # رسم النص العربي (المصلح) والمسافة
158                 display_str = f"{arabic_name} | {dist_cm} سم"
159                 frame = draw_arabic_text(frame, display_str, (new_x, y - 40))
160 
```

1.حلقة المعالجة والفلترة (الأسطر 137 - 140)

- **if len(indexes) > 0:** يتأكد البرنامج أولاً أنه وجد أجساماً بالفعل قبل البدء في الرسم.
- **for i in indexes.flatten():** يمر على كل جسم نجح في اختبار الـ NMS أي الأجسام الحقيقية فقط بدون تكرار
- **if label in target_objects:** هنا تتم الفلترة النهائية؛ فبالرغم من أن YOLO قد يرى "أريكة" أو "طاولة"، إلا أنه لن ينفذ الأوامر التالية إلا إذا كان هذا الجسم موجوداً في قائمة الاهتمامات التي حددتها أنت في بداية الكود.

2.حساب المسافة رياضياً (الأسطر 143 - 146)

هذا الجزء هو تطبيق عملي لقانون **التشابه المثلثي** (Triangle Similarity)لحساب بُعد الأجسام:

- **known_w = KNOWN_WIDTHS.get(label, 15):** يجلب العرض الحقيقي للجسم من القاموس الذي أعددناه (وإذا لم يجد العرض، يفترض أنه 15 سم كقيمة افتراضية).
- **معادلة المسافة (السطر 145):**

$$distance = \frac{known\_w \times FOCAL\_LENGTH}{w}$$

حيث wهو عرض الجسم بالبكسل كما تراه الكاميرا. كلما صغر الجسم في الصورة، زادت المسافة المحسوبة.

3.تحضير جملة النطق والترجمة (الأسطر 148 - 149)

- **arabic_name**:يبحث عن اسم الجسم بالعربي في الـ translation_map.
- **detected_for_speech.append(...)**:يقوم بتركيب جملة كاملة بالعربية مثل" **كوب على بعد 50 سنتيمتر** "ويضيفها لقائمة الانتظار ليتم نطقها لاحقاً.

4. تعديل الإحداثيات "تأثير المرآة" (السطر 151 - 152)

- **new_x = width - (x + w)**:هذه لمسة ذكية؛ فغالباً ما تكون الكاميرات الأمامية "مقلوبة " (Mirror effect).هذا السطر يعكس إحداثيات المربع لكي يتطابق موقع المربع مع موقع الجسم الحقيقي بالنسبة للمستخدم.

5. الرسم وكتابة النص العربي (الأسطر 154 - 159)

- **رسم المربع (السطر 155)** cv2.rectangle **:**يرسم إطاراً أخضر اللون (0, 255, 0) حول الجسم المكتشف.
- **تحضير النص الظاهر** display_str **:**يدمج اسم الجسم بالمسافة في نص واحد (مثل: "كوب | 50 سم").
- **رسم النص بالعربي (السطر 159) :**يستدعي الدالة draw_arabic_textالتي شرحناها سابقاً لوضع النص فوق المربع بشكل صحيح ومنسق، باستخدام إحداثيات تبعد 40 بكسل فوق المربع لضمان الوضوح.

النتيجة النهائية لهذا الكود:

عندما يرى البرنامج "زجاجة" مثلاً، سيقوم في أجزاء من الثانية بـ:

1. تحديد مكانها.
2. حساب بُعدها بالسنتيمتر.
3. رسم مربع أخضر حولها.
4. كتابة "**زجاجة | 70 سم** "بالعربية فوقها.
5. تجهيز صوت يقول" :**زجاجة على بعد 70 سنتيمتر.**"

```python
# 5. واجهة Tkinter
# ============================================================
root = tk.Tk()
root.title("👀 مساعد الرؤية الذكي")
root.geometry("800x680")
root.configure(bg="#1a1a1a")

video_label = tk.Label(root, bg="black")
video_label.pack(pady=10)

status_label = tk.Label(root, text="⏳ جارٍ التحميل...", font=("Arial", 14), fg="yellow", bg="#1a1a1a")
status_label.pack(pady=10)

cap = None

def start_camera():
    global cap
    cap = cv2.VideoCapture(1)
    if not cap.isOpened():
        status_label.config(text="❌ الكاميرا غير متصلة", fg="red")
        return
    status_label.config(text="✅ النظام جاهز - القياس والنطق مفعل", fg="#00ff00")
    update_frame()
```

1.هيكل النافذة(Tkinter Setup)

في الصورة الثالثة (الأسطر 171 - 182)، نقوم بإنشاء التصميم:

- **root = tk.Tk():** إنشاء النافذة الرئيسية للبرنامج.
- **root.title ("مساعد الرؤية الذكي"):** وضع عنوان للنافذة يظهر في الأعلى.
- **root .geometry ("800x680")** تحديد أبعاد النافذة (العرض والطول).
- **video_label** هذا هو "الإطار الأسود" الذي سيتم عرض بث الكاميرا داخله. تم ضبط خلفيته باللون الأسود bg="black".
- **status_label** نص تفاعلي يخبر المستخدم بما يحدث الآن (مثلاً: "⏳ جارٍ التحميل...").

2. إدارة الكاميرا(Camera Control)

في الأسطر من 186 إلى 193:

- **def start_camera()**: دالة تبدأ العمل عند تشغيل البرنامج.
- **cap = cv2.VideoCapture(1)**: محاولة فتح الكاميرا. الرقم 1 يشير عادةً إلى كاميرا خارجية (USB)، وإذا كنت تستخدم كاميرا اللابتوب المدمجة فقد تحتاج لتغييره إلى 0.
- **فحص الاتصال (الأسطر 189-191):** إذا فشل البرنامج في العثور على كاميرا، يتغير نص status_label إلى اللون الأحمر مع رسالة ❌**"الكاميرا غير متصلة."**
- **النجاح:** إذا فُتحت الكاميرا بنجاح، تظهر رسالة خضراء ✅**"النظام جاهز "**ويبدأ استدعاء دالة update_frame() لتحديث الصور باستمرار.

3. منطق القياس والذكاء الاصطناعي (الصور 1 و 2)

- **حساب المسافات (الأسطر 44-45):** يستخدم الكود KNOWN_WIDTHS و FOCAL_LENGTH لتطبيق معادلة رياضية تحول حجم الجسم في الصورة إلى مسافة حقيقية بالسنتيمتر.
- **الفلاتر (الأسطر 47-48):** CONFIDENCE_THRESHOLD: * لضمان أن البرنامج لا يخطئ في تشخيص الأجسام (يتجاهل أي شيء يشك فيه بنسبة أقل من 50%).
    - NMS_THRESHOLD: لمنع ظهور مربعات كثيرة متداخلة على نفس الجسم.

- **الترجمة العربية (الأسطر 27-41):** قاموس translation_map يحول مخرجات YOLO مثل 'cup') إلى كلمات عربية ('كوب') لتظهر للمستخدم بشكل مفهوم.

4. نظام النطق الذكي(Speech Management)

في الأسطر 49 إلى 52:

- **SPEECH_DELAY = 5:** يمنع البرنامج من "الثرثرة" الزائدة؛ حيث لن ينطق اسم الجسم مرة أخرى إلا بعد مرور 5 ثوانٍ على الأقل.
- **is_speaking:** " مفتاح أمان" يضمن أن البرنامج لا يحاول نطق جملتين في نفس اللحظة، مما يمنع تداخل الأصوات وتجمد البرنامج.

ملخص سير العمل في هذا الجزء:

1. يفتح البرنامج نافذة سوداء أنيقة.
2. يبحث عن الكاميرا الخارجية.
3. بمجرد العثور عليها، يبدأ في سحب الصور وإرسالها لنموذج **YOLO**.
4. يترجم النتائج للعربية، يحسب المسافة، ثم يعرض كل ذلك داخل الـ video_label في النافذة.

```
194
195 def update_frame():
196     if cap and cap.isOpened():
197         ret, frame = cap.read()
198         if ret:
199             frame = process_frame(frame)
200             rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
201             img = ImageTk.PhotoImage(Image.fromarray(rgb))
202             video_label.config(image=img)
203             video_label.image = img
204     root.after(20, update_frame)
205
206 def on_close():
207     if cap: cap.release()
208     root.destroy()
209
210 root.after(500, start_camera)
211 root.protocol("WM_DELETE_WINDOW", on_close)
212 root.mainloop()
```

1. دالة تحديث الإطارات update_frame()

هذه الدالة (الأسطر 195-204) هي المسؤولة عن جعل الفيديو يبدو مستمراً وليس مجرد صورة ثابتة:

- **قراءة الكاميرا cap.read():** تأخذ اللقطة الحالية من عدسة الكاميرا.
- **المعالجة الذكية frame = process_frame(frame):** ترسل الصورة الخام إلى "المصنع" الذي شرحناه سابقاً (YOLO) ليرسم المربعات، يحسب المسافات، ويترجم الأسماء للعربية.
- **تحويل الألوان cv2.cvtColor(frame, cv2.COLOR_BGR2RGB):** ضروري لأن OpenCV يقرأ الألوان بترتيب (أزرق-أخضر-أحمر)، بينما واجهة Tkinter تفهمها بترتيب (أحمر-أخضر-أزرق).
- **العرض على الواجهة:** يتم تحويل الصورة المعالجة إلى تنسيق PhotoImage ووضعها داخل الـ video_label ليراها المستخدم.

2. دالة الإغلاق الآمن on_close()

الأسطر 206-208 تضمن عدم "تعليق" الكاميرا بعد قفل البرنامج:

- **تحرير الكاميرا** cap.release() **:**تخبر نظام التشغيل أن البرنامج انتهى من استخدام الكاميرا، لكي تتمكن برامج أخرى) مثل Zoom أو (Skype)من استخدامها لاحقاً.
- **تدمير النافذة** root.destroy() **:**يغلق واجهة المستخدم ويمسحها من الذاكرة العشوائية.

3. إطلاق البرنامج(The Launchpad)

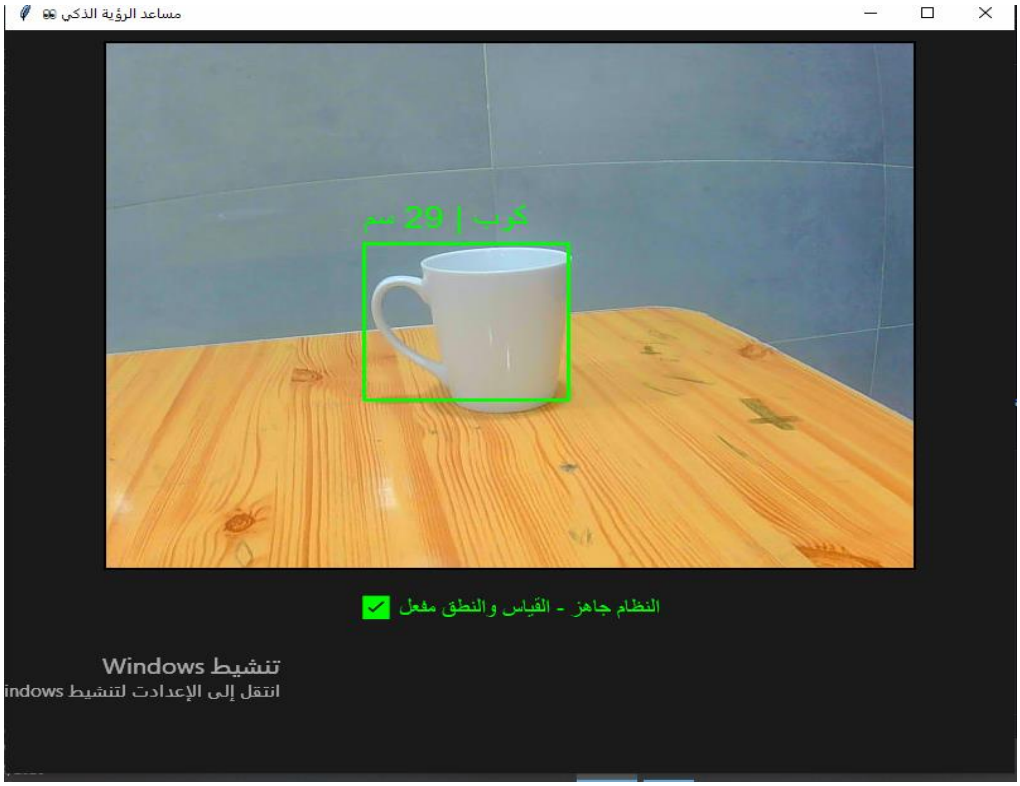
الأسطر من 210 إلى 212 هي "مفتاح التشغيل:"

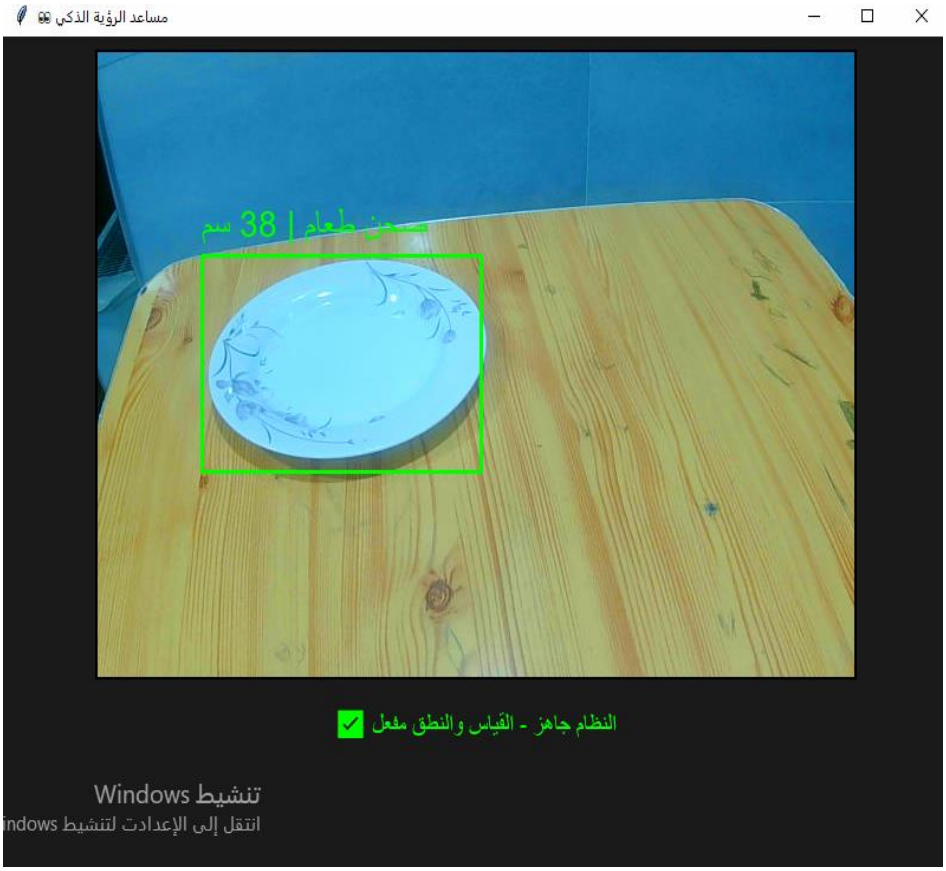
- **root.after(500, start_camera)**:ينتظر نصف ثانية بعد فتح النافذة ثم يشغل الكاميرا. هذا التأخير البسيط يضمن أن الواجهة الرسومية استقرت تماماً قبل البدء في معالجة الفيديو الثقيلة.
- **root.protocol(...)**:يربط زر "X" الموجود في زاوية النافذة بدالة on_close لضمان الإغلاق الآمن.
- **root.mainloop()**:هذه هي الدالة التي تمنع البرنامج من الانغلاق فور تشغيله؛ فهي تجعل النافذة "تنتظر" تفاعلات المستخدم (أوامر الكاميرا، الضغط على الأزرار، إلخ).

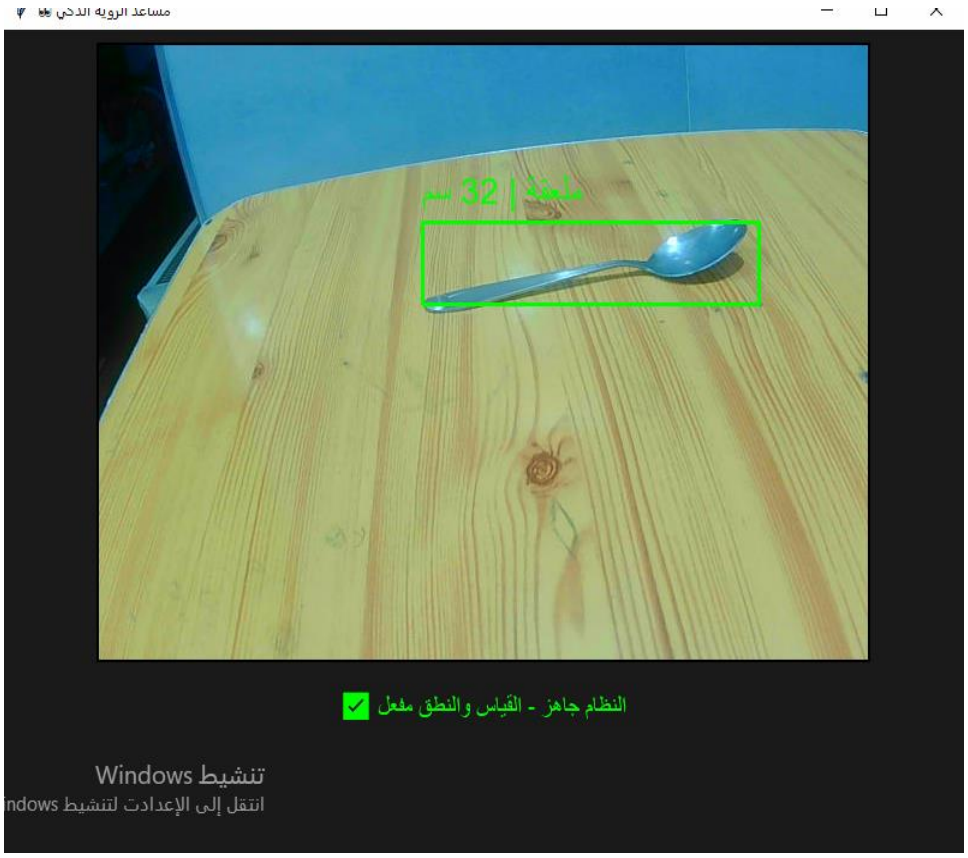
ملخص رحلة البيانات داخل الكود (من البداية للنهاية):

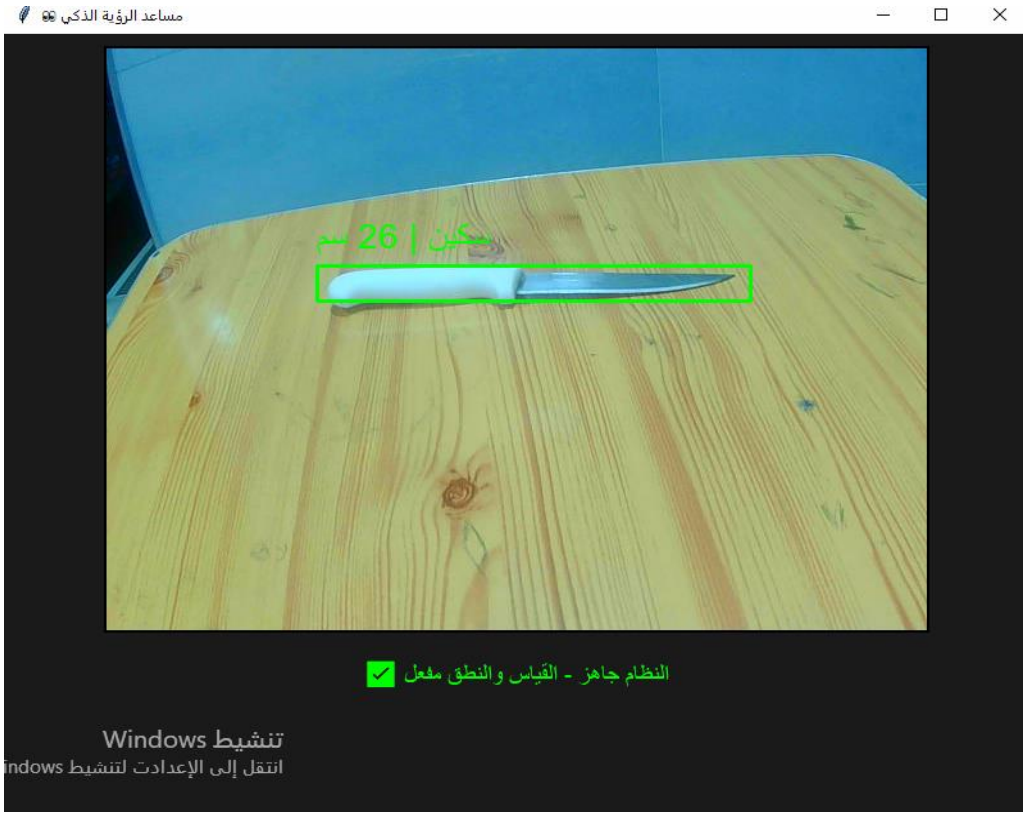
1. **البداية :**تفتح نافذة Tkinter وتبحث عن الكاميرا.
2. **المدخلات :**تسحب دالة update_frame صورة من الكاميرا.
3. **المعالجة :**تذهب الصورة لـ(YOLO) باستخدام الأوزان والملفات التي حُملت في البداية لتحديد الأشياء.
4. **الحساب :**يتم حساب المسافة بناءً على عرض الجسم بكسل مقابل .KNOWN_WIDTHS
5. **المخرجات الرسومية :**يُرسم المربع والنص العربي (draw_arabic_text) على الصورة.
6. **المخرجات الصوتية :**إذا مرّت 5 ثوانٍ، تنطق دالة speak_text اسم الجسم والمسافة.
7. **التكرار :**تعود الدورة للخطوة رقم 2 بسرعة فائقة.

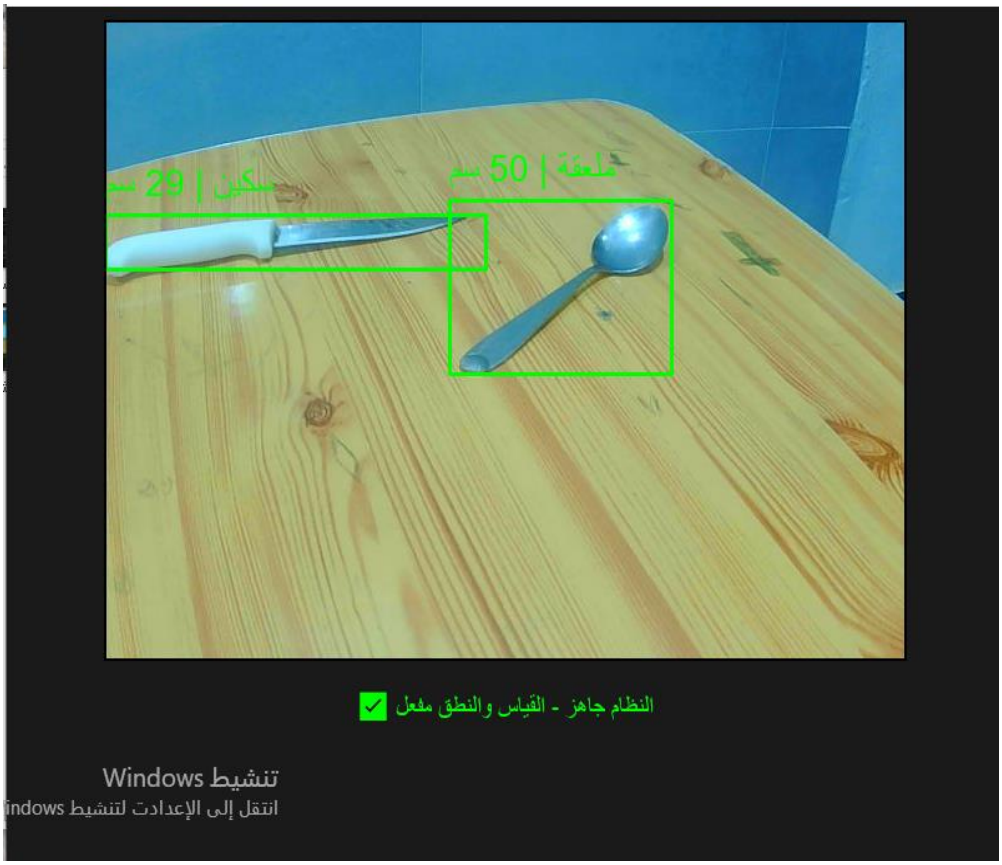
التنفيذ


مساعد الرؤية الذكي
كوب | 29 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط Windows

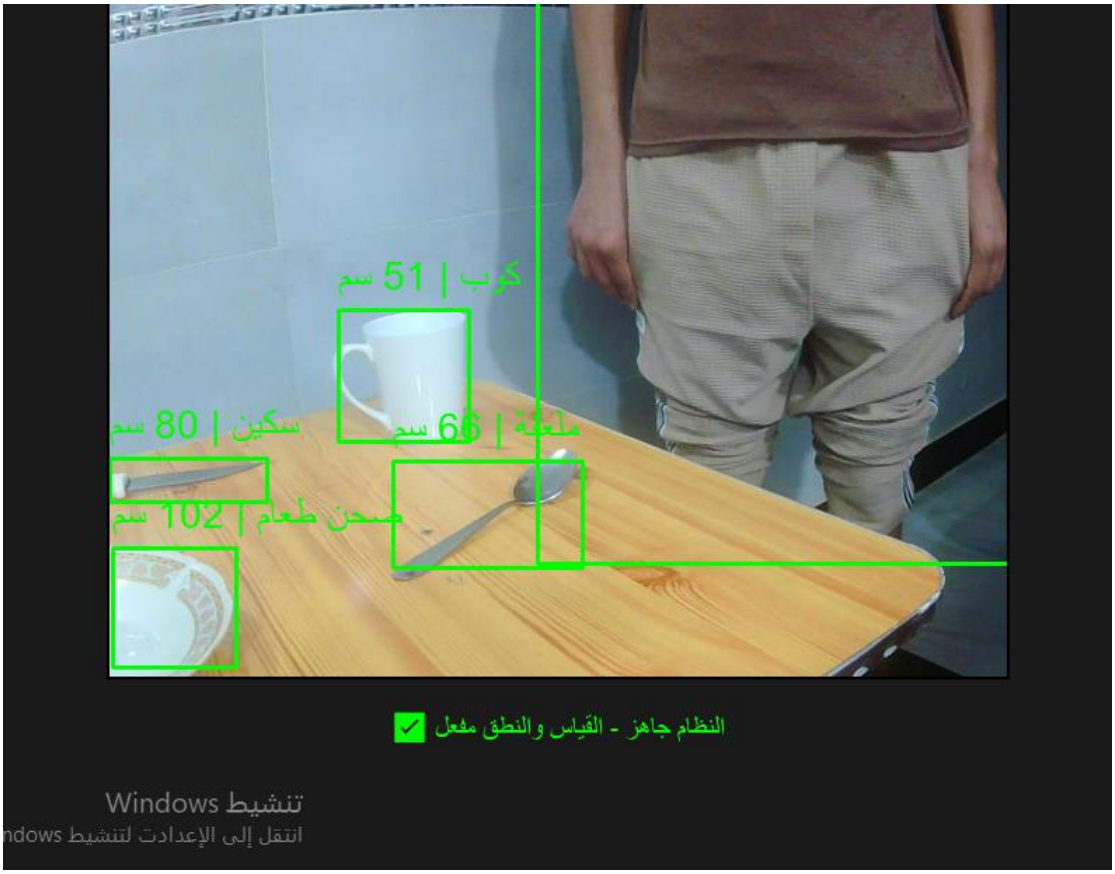

مساعد الرؤية الذكي
صحن طعام | 38 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط Windows


مساعد الرؤية الذكي
ملعقة | 32 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط Windows


مساعد الرؤية الذكي
سكين | 26 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط Windows


ملعقة | 50 سم
سكين | 29 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط indows


سكين | 36 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادت لتنشيط indows


كوب | 51 سم
ملعقة | 66 سم
سكين | 80 سم
صحن طعام | 102 سم
النظام جاهز - القياس والنطق مفعل
تنشيط Windows
انتقل إلى الإعدادات لتنشيط ndows

الفصل الخامس

الاستنتاجات والمقترحات

1.5 الاستنتاجات

1.الاستنتاج الوظيفي (ماذا يقدم النظام؟)

هذا الكود يحول الكاميرا من "عين رقمية" تسجل الفيديو فقط إلى مترجم للواقع حيث يقوم بـ:

- **الإدراك المكاني :**التعرف على ماهية الأجسام (كوب، شخص، سيارة) وسط بيئة معقدة.
- **الوعي بالمسافات :**تقدير البُعد الفعلي للأشياء عن المستخدم بدقة السنتيمتر، مما يساعد في تجنب العقبات.
- **التواصل الصوتي :**إعطاء تنبيهات صوتية باللغة العربية، مما يجعله تطبيقاً "تفاعلياً" لا يحتاج من المستخدم النظر إلى الشاشة.

2. التكامل التقني (كيف تم الربط؟)

الاستنتاج المثير هنا هو كيفية دمج خمس مكتبات ضخمة لتعمل بانسجام في حلقة زمنية لا تتعدى 20 ميلي ثانية:

- **OpenCV:**للتعامل مع الفيديو والذكاء الاصطناعي.(YOLO)
- **PyTorch/DNN:**لإدارة الحسابات المعقدة للشبكة العصبية.
- **gTTS:**للربط مع خوادم جوجل السحابية لتحويل النص إلى كلام بلكنة طبيعية.
- **Tkinter:**لتوفير واجهة مستخدم بسيطة ومستقرة.
- **Pillow (PIL):**لحل أعقد مشاكل OpenCV وهي دعم اللغة العربية.

3. معالجة التحديات (الذكاء البرمجي)

يظهر الكود استنتاجاً ذكياً في حل مشكلات تقنية شائعة:

- **مشكلة "الثرثرة :"**تم حلها باستخدام SPEECH_DELAY وis_speaking لضمان نطق المعلومات الهامة فقط دون تكرار مزعج.
- **مشكلة "اللغة العربية :"**تم حلها بإعادة صياغة النصوص وعكسها برمجياً لتظهر بشكل صحيح (Reshaping & Bidi).
- **مشكلة "المربعات المتعددة :"**تم حلها باستخدام تقنية NMS لضمان الدقة وتوفير موارد المعالج.

4. العمل المقترح

الهدف النهائي من هذا العمل هو تعويض حاسة البصر أو تعزيزها .يمكن تثبيت هذا الكود على:

1. **نظارات ذكية :**تنبه الكفيف بما يحيط به.
2. **روبوتات الخدمة :**لتحديد الأشياء المطلوبة (مثلاً: "أحضر الكوب الذي يبعد 50 سم").
3. **أنظمة المراقبة الذكية :**التي لا تكتفي بتصوير اللصوص، بل تحدد هويتهم وبُعدهم عن الهدف.

المراجع

1. World Health Organization (WHO), “Blindness and Vision Impairment,” who.int.**https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment** (accessed on 31 March 2026).

2. N. H. Ismail and M. N. A. Rahman, "Challenges and barriers of public transportation for people with visual impairment: A review," *Journal of Transport & Health*, 2020.

3. S. K. S. Tyagi et al., "Assistive technologies for visually impaired: A review," *IEEE Access*, 2021.

4. R. S. S. Kumari and S. Bharathi, "Bus identification system for visually impaired people using RFID," *International Journal of Computer Applications*, 2014. *of Engineering and Applied Sciences*, 2018.

5. P. PA war and S. G. Sankarand, "A smart bus station for the visually impaired using RFID and wireless sensor networks," *International Journal of Pervasive Computing and Communications*, 2016.

6.Szeliski, Richard. (2022). *Computer Vision: Algorithms and Applications*. Second Edition. Springer Nature.

7. CS231n (Stanford University**)**: "Convolutional Neural Networks for Visual Recognitio.

8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press

9. Stanford University CS231n: Convolutional Neural Networks for Visual Recognition.

10. Szeliski, *R. (2022).* Computer Vision: Algorithms and Applications. *Springer Nature.*

11.: Unified, Real-Time Object Detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR).

12.*Stanford University.* "Detection and Segmentation". cs231n.stanford.edu

13. Le n, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.Cu

14 Szeliski, R. (2022**).** Computer Vision: Algorithms and Applications. Second Edition. Springer Nature.

15. Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. CVPR.

16. Dalal, N., & Triggs, B. (2005). Histograms of Oriented Gradients for Human Detection. CVPR.

17. Redmon, J., et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. (الورقة الأساسيةYOLO).

18. Girshick**,** R. (2015). Fas*t* R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV).

19. Harris, C. R., et al. (2020). Array programming with NumPy. Nature.

20.*Paszke, A., et al. (2019).* PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems.

21. Szeliski, R. (2022). Computer Vision: Algorithms and Applications.

22. Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library.