Nestkastdetectie Dashboard – Samenvatting

Inhoud

[Overzicht van het project en architectuur 1](#_Toc202531210)

[Belangrijke bestanden 1](#_Toc202531211)

[Ondersteunende bestanden: 1](#_Toc202531212)

[Van afbeelding naar nestlocaties: de analysepipeline 1](#_Toc202531213)

[Integratie in Streamlit 2](#_Toc202531214)

[Alternatieve modellen 2](#_Toc202531215)

[Waarom YOLOv8? 2](#_Toc202531216)

[Onderhoud en uitbreiding 2](#_Toc202531217)

[Samenvatting van de logica 2](#_Toc202531218)

[De logica werkt als volgt: 3](#_Toc202531219)

[Aanpassen van de logica: 3](#_Toc202531220)

[Aanbevelingen voor verbetering: 3](#_Toc202531221)

Overzicht van het project en architectuur  
De toepassing is gebouwd als een Streamlit-dashboard waarin gebruikers een diersoort kiezen (huismus, gierzwaluw of vleermuis) en een gevelafbeelding uploaden. Het systeem detecteert geschikte nestlocaties en toont deze visueel en als tekst.

Belangrijke bestanden:  
- main.py: Streamlit-interface; koppelt gebruikersinput aan de analyse.  
- yolo\_model/nestlogica.py: Kern van de analyse; gebruikt YOLOv8 voor detectie en past logica toe voor filtering en selectie van nestplaatsen.  
- model\_predict.py: Hulpscript voor losse modeltests, niet essentieel voor de dashboardflow.  
- Getraind model (yolov8n\_nest\_50epochs.pt): YOLOv8 nano-model getraind op gevelafbeeldingen.

Ondersteunende bestanden:  
Bevatten dependencies (environment.yaml), notebooks voor modeltraining en experimenten, en testbestanden.

Van afbeelding naar nestlocaties: de analysepipeline  
1. YOLO-predictie: Detecteert gevel en ramen via bounding boxes en segmentatiemaskers.  
2. Maskergeneratie: Op basis van de predictie worden gevel- en raamgebieden als binaire maskers opgeslagen.  
3. Filtering:  
 - 0,6 m van gevelranden;  
 - ≥1,0 m afstand van ramen;  
 - hoogtebeperkingen per diersoort (bijv. 3–10 m voor huismus).  
4. Selectie nestlocaties: Maximaal 5 locaties, met minimaal 3 m afstand onderling.  
5. Visualisatie: De inputafbeelding met:  
 - groen gebied (geschikte pixels);  
 - soortspecifieke icoontjes op geselecteerde locaties;  
 - bijbehorende X,Y-coördinaten en hoogte in meters.

**Afbeelding met tekst, schermopname, diagram, lijn

Door AI gegenereerde inhoud is mogelijk onjuist.**

Integratie in Streamlit  
De analyse wordt geactiveerd via een knop in de UI. main.py roept analyseer\_nestlocaties() aan in nestlogica.py, dat alle stappen uitvoert en de resultaten visualiseert. Model wordt per analyse opnieuw geladen (voor eenvoud), maar dit kan geoptimaliseerd worden. Alle logica (filters, hoogtecriteria) zit vast in de code, maar kan in de toekomst parametriseerbaar worden gemaakt via sliders of invoervelden.

Alternatieve modellen  
Tijdens het project zijn ook andere methodes onderzocht. Deze zijn uitgewerkt in aparte branches van het project:  
- OpenCV-gebaseerde klassieke beeldverwerking: Contourdetectie, vormfilters en randverwerking. Werkte in eenvoudige gevallen, maar was niet robuust genoeg.  
- Segmentatie met SAM (Segment Anything Model): Segmenten werden automatisch gegeneerd, maar vereisten veel logica om relevante objecten te isoleren. SAM is krachtig, maar zwaarder en minder geschikt voor real-time dashboardgebruik.

Waarom YOLOv8?  
- Betrouwbaarder dan klassieke filters of SAM bij variërende geveltypes.  
- Eenvoudige integratie via Ultralytics API.  
- Schaalbaar en uitbreidbaar naar andere klassen of diersoorten.  
- Snelheid: YOLOv8n is lichtgewicht en geschikt voor livegebruik in een Streamlit-app.

Onderhoud en uitbreiding  
Kritieke onderdelen:  
- Modelbestand: Hertrainen vereist bij detectie van nieuwe objecten.  
- nestlogica.py: Aanpassingen in logica en filtering gebeuren hier.  
- main.py: Voor uitbreidingen van UI en gebruikersinteractie.  
- Notebooks (in branches): Nuttig voor modeltraining of analyse van detectiefouten.

# Samenvatting van de logica

Het nestkastdetectiesysteem werkt met een zogeheten "pixelmasker", waarin per pixel van de afbeelding wordt bepaald of deze geschikt is voor een nestkast. Dit gebeurt op basis van YOLOv8-detectie, logische regels en afstandsberekeningen.

De logica werkt als volgt:  
1. Een getraind YOLOv8-model detecteert ramen en gevels op de afbeelding.  
2. Op basis hiervan worden twee binaire maskers opgebouwd:  
 - gevelmasker (pixels van het gebouw)  
 - raam/deurmasker (pixels met ramen/deuren)  
3. Vervolgens worden alleen die gevelpixels toegestaan die:  
 - minstens 60 cm van de rand van de gevel liggen;  
 - minimaal 1 meter van een raam verwijderd zijn;  
 - binnen het hoogte-interval vallen dat hoort bij de gekozen diersoort (bijv. 3–10 m voor een huismus).  
4. Het resultaat is een “allowed\_mask”: een kaart van alle pixels waar een nestkast zou mogen komen.  
5. Hieruit worden maximaal 5 locaties gekozen die onderling ≥3 meter uit elkaar liggen.  
6. Deze locaties worden gevisualiseerd op de afbeelding met groene gebieden (toegestane pixels) en icoontjes van de soort.

Aanpassen van de logica:  
- Afstanden (zoals marge of afstand tot ramen) zijn in meters instelbaar in de code via vaste waarden.  
- Hoogte-eisen per soort staan bovenin het script als dictionary.  
- Het maximaal aantal nesten en de minimale onderlinge afstand zijn instelbaar bij de selectie.

Aanbevelingen voor verbetering:  
- Laat gebruikers zelf parameters instellen via de interface (bijv. sliders voor afstand of hoogte).  
- Maak de schaal (PIXEL\_SCALE) dynamisch in plaats van hard-coded, zodat deze per afbeelding kan verschillen.