In gewone mensentaal werkt je project nu zo: je start met main.py, dat een PyQt5–applicatie laadt waarmee je een Baumer-camera (via NeoAPI/GenICam) kunt verbinden, configureren en bedienen. Instellingen (Exposure, Gain, Flash, etc.) sla je als named profiles op in een SQLite-database. In je GUI kun je live-beelden zien, software-triggers en flash-cycli aansturen, en beelden opslaan.

Wat je nog wilt toevoegen:

* **Point Selector**: met een knop in de hoofd-GUI een aparte module openen waarin je handmatig punten op een testbeeld selecteert (en opslaat als JSON).
* **Inspect Points**: na elke snapshot direct een automatische beeldinspectie uitvoeren op de laatst genomen foto, binnen een ROI en op basis van intensiteits-drempels, en de resultaten in de GUI tonen.

**Projectoverzicht**

* **Entry-point**: main.py – start van je PyQt5-applicatie voor Baumer-camera aansturing.
* **Virtuele omgeving**: staat buiten je project (C:\virt omgeving\AI camera) omdat je code in OneDrive bewaart om naadloos op meerdere computers te werken.
* **Structuur**:

css

KopiërenBewerken

baumer\_project/

├─ main.py

├─ \*.bat (backup, start…)

├─ cameras/ (camera\_wrapper.py)

├─ config/ (config\_manager.py + settings.db)

├─ gui/ (main\_window.py, settings\_dialog.py, plugins/)

├─ processing/ (point\_selector.py, inspect\_points.py)

├─ resources/ (GenICam-XML, icons, json\_points, styles)

└─ oude scripts/ (legacy-code)

* **Integratie-tools**:
  + **ConfigManager** laadt config/settings.db automatisch via os.path.dirname(\_\_file\_\_).
  + **Point Selector** en **Inspect Points** zijn nu modules in processing/ met functie-API’s (run\_point\_selector(), inspect\_points(image, params)).

**Verbeter- en uitbreidingsvoorstellen**

1. **Geavanceerde camera-features**
   * **Chunk-data**: actief gebruik van GenICam ChunkEnable om metadata (FrameID, Timestamp, Gain, ExposureTime, …) mee te sturen in elk frame [media.baumer.com](https://media.baumer.com/Baumer_TDS_VAX-50C.I.NX16_EN_20240306_DS_11726725.pdf?utm_source=chatgpt.com).
   * **Hardware-trigger & sequencer**: FPGA-gebaseerde triggering en sequenties bij VAX-50C.I.NX16, voor lage jitter en deterministische acquisitie [media.baumer.com](https://media.baumer.com/Baumer_TDS_VAX-50C.I.NX16_EN_20240306_DS_11726725.pdf?utm_source=chatgpt.com).
2. **GPU-versnelde beeldverwerking (OpenCV)**
   * **OpenCL-UMat**: zet cv2.Mat om naar cv2.UMat om standaard OpenCV-functies transparant op de GPU te laten draaien via OpenCL [answers.opencv.org](https://answers.opencv.org/question/232802/accelerate-opencv-functions-on-gpu-through-cuda/?utm_source=chatgpt.com).
   * **Lens-kalibratie & distortie-correctie**: gebruik de OpenCV-calibratiemodule (calibrateCamera, undistort) om lensvervorming te corrigeren in je snapshots [docs.opencv.org](https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html?utm_source=chatgpt.com).
3. **Deep-learning integratie**
   * **YOLO-objectdetectie**: real-time detectie van objecten of defecten met YOLO-modellen binnen OpenCV, voor automatische kwaliteitsinspectie [GeeksforGeeks](https://www.geeksforgeeks.org/computer-vision/object-detection-with-yolo-and-opencv/?utm_source=chatgpt.com).
4. **Schaalbaarheid & UX**
   * **Multithreading-pipeline**: capture, processing en UI-update in gescheiden threads om haperingen te voorkomen.
   * **Remote monitoring**: RTSP-stream of web-interface voor extern live-beeld.
   * **Multi-camera**: synchronisatie via IEEE 1588 voor stereo-of 3D-toepassingen.

Met deze verbeteringen gebruik je de full-power van je Baumer-hardware en OpenCV om van een handmatig captureren naar een volledig geautomatiseerde, real-time inspectie-applicatie te gaan.