ROIDS - Region Of Interest Designation System

Jason Jenkins

2025-11-15

Contents

1 Project Overview

ROIDS (Region Of Interest Designation System) is a cross-platform native desktop application designed to help users define polygons and lines on images and video frames. These geometric annotations serve as configuration inputs for computer vision systems that count people or other objects within designated regions or crossing specified lines.

1.1 Purpose

The application enables users to:

- Visually define regions of interest (polygons) and counting lines on media files
- Export geometric data in a normalized, machine-readable format
- Iterate on configurations by reloading and editing previous work

1.2 Target Platforms

Priority order:

1. macOS (primary target)
2. Linux (secondary target)
3. Windows (nice-to-have)

2 Goals and Objectives

2.1 Primary Goals

- Provide an intuitive interface for defining polygons and lines on static images and video frames

- Export geometric data in normalized coordinates (0.0 to 1.0 range) suitable for CV system configuration
- Enable iterative refinement through reload and edit capabilities
- Maintain simplicity and focus on core functionality for MVP

2.2 Non-Goals (For MVP)

- Real-time video processing or analysis
- Integration with computer vision processing pipelines
- Advanced styling or visualization options
- Multi-user or collaborative editing
- Cloud storage or synchronization

3 User Stories

3.1 Core User Journey

1. As a user, I want to select an image or video file from my filesystem
2. As a user, I want to see the selected media displayed in the application window
3. As a user, I want to draw polygons and lines using mouse clicks
4. As a user, I want to edit the positions of vertices and endpoints
5. As a user, I want to name my regions and lines
6. As a user, I want to export my annotations to a YAML or JSON file
7. As a user, I want to reload a media file and its associated annotation file to make edits

3.2 Specific Interactions

- As a user working with video, I want to scrub through the timeline to select an appropriate frame for annotation
- As a user, I want to delete entire shapes or individual vertices
- As a user, I want to close a polygon or leave it as an open polyline
- As a user, I want my annotations to have sensible default names that I can optionally customize

4 Functional Requirements

4.1 FR-001: Media File Selection

- The application SHALL provide a native file selection dialog
- The application SHALL support common image formats (JPEG, PNG, BMP, TIFF)
- The application SHALL support common video formats (MP4, AVI, MOV, MKV)
- The application SHALL validate that selected files are readable and supported

4.2 FR-002: Image Display

- The application SHALL display the selected image in the main window
- The application SHALL scale the image to fit the window while maintaining aspect ratio
- The application SHALL allow zooming and panning (nice-to-have for MVP)

4.3 FR-003: Video Frame Selection

- The application SHALL provide a timeline scrubber control for video files
- The application SHALL allow users to select a specific frame for annotation

- The application SHALL assume fixed camera position (ROIs apply to entire video)
- The application SHOULD display the current frame number or timestamp
- Video playback with ROI overlay is a nice-to-have feature (not required for MVP)

4.4 FR-004: Polygon Creation

- The application SHALL allow users to create polygons by clicking to add vertices
- The application SHALL close a polygon when the user double-clicks
- The application SHALL allow users to cancel polygon creation with the Escape key (creating an open polyline)
- The application SHALL visually indicate the polygon being created
- The application SHALL have no practical limit on the number of vertices

4.5 FR-005: Line Creation

- The application SHALL allow users to create lines (two-point segments)
- The application SHALL support polylines (multi-segment lines)
- The application SHALL treat lines as unclosed polygons for implementation purposes

4.6 FR-006: Annotation Editing

- The application SHALL allow users to select existing polygons and lines
- The application SHALL allow users to drag vertices and endpoints to new positions
- The application SHALL provide visual feedback for selected shapes and vertices

- The application SHALL allow deletion of entire shapes via the Delete key
- The application SHALL allow deletion of individual vertices via Ctrl+Click

4.7 FR-007: Annotation Naming

- The application SHALL assign default sequential names ("region 1", "region 2", "line 1", etc.)
- The application SHALL allow users to optionally rename any annotation
- The application SHALL ensure names are preserved in exported data

4.8 FR-008: Data Export

- The application SHALL export annotations to YAML or JSON format
- The application SHALL normalize all coordinates to floating-point values between 0.0 and 1.0
- The application SHALL include annotation names/labels in the export
- Normalized coordinates SHALL be relative to the original media dimensions
- The export format SHALL be human-readable and machine-parseable

4.9 FR-009: Reload and Edit

- The application SHALL allow users to load a previously exported annotation file
- The application SHALL associate the annotation file with its corresponding media file
- The application SHALL reconstruct all polygons and lines from the loaded data
- The application SHALL allow editing of reloaded annotations

4.10 FR-010: Basic UI Operations

- The application SHALL provide menu or toolbar access to key functions
- The application SHALL indicate the current tool/mode to the user
- The application SHALL handle window resizing gracefully
- The application SHOULD provide keyboard shortcuts for common operations

5 Non-Functional Requirements

5.1 NFR-001: Performance

- The application SHALL load and display images up to 4K resolution without significant delay
- The application SHALL handle video files up to 1 hour in length
- The application SHALL respond to user interactions within 100ms under normal conditions

5.2 NFR-002: Usability

- The application SHALL use native-looking UI widgets and dialogs
- The application SHALL follow platform conventions for keyboard shortcuts
- The application SHALL provide visual feedback for all user actions
- Error messages SHALL be clear and actionable

5.3 NFR-003: Reliability

- The application SHALL validate all user inputs
- The application SHALL handle file I/O errors gracefully
- The application SHALL prevent data loss during normal operation

5.4 NFR-004: Maintainability

- Code SHALL be well-documented with clear comments
- The application SHALL follow established design patterns
- Dependencies SHALL be minimized and well-justified

6 Technical Requirements

6.1 TR-001: Programming Language and Framework

Recommended technology stack:

- **Language:** Python 3.10+
- **GUI Framework:** PyQt6 or PySide6 (Qt for Python)
- **Video Processing:** OpenCV (cv2) for video frame extraction
- **Image Handling:** Pillow (PIL) for image loading and manipulation
- **Data Serialization:** PyYAML for YAML, built-in json module for JSON

Rationale:

- Python provides rapid development and you have strong experience
- Qt provides native-looking widgets on all target platforms
- Qt handles cross-platform differences without platform-specific code
- OpenCV is industry-standard for video/image processing
- Mature ecosystem with excellent documentation

6.2 TR-002: Data Format Specification

6.2.1 Export Format Structure (YAML Example)

```
media_file: "path/to/video.mp4"
frame_width: 1920
frame_height: 1080
frame_number: 150  # for videos only
annotations:
```

```
  - name: "region 1"
    type: "polygon"
    vertices:
      - [0.1234, 0.2345]
      - [0.3456, 0.2345]
      - [0.3456, 0.6789]
      - [0.1234, 0.6789]
  - name: "line 1"
    type: "line"
    vertices:
      - [0.5000, 0.0000]
      - [0.5000, 1.0000]
```

6.2.2 Coordinate Normalization

- X coordinates: pixel_x / image_{width} → [0.0, 1.0]
- Y coordinates: pixel_y / image_{height} → [0.0, 1.0]
- Origin: top-left corner (0.0, 0.0)
- Bottom-right corner: (1.0, 1.0)

6.3 TR-003: Project Structure

```
roids/
 README.org
 requirements.txt
 setup.py
 src/
    __init__.py
    main.py              # Application entry point
    ui/
       __init__.py
       main_window.py   # Main application window
       canvas.py        # Drawing canvas widget
       controls.py      # Timeline, toolbars, etc.
    models/
       __init__.py
       annotation.py    # Annotation data structures
       project.py       # Project state management
```

```
   io/
      __init__.py
      media_loader.py  # Image/video loading
      exporter.py      # YAML/JSON export
   utils/
       __init__.py
       geometry.py      # Coordinate transformations
tests/
   ...
docs/
    project.org          # This file
```

6.4 TR-004: Development Environment

- Version control: Git
- Dependency management: pip + requirements.txt or poetry
- Testing framework: pytest
- Code formatting: black, isort
- Linting: pylint or ruff

7 UI/UX Specifications

7.1 Main Window Layout

```
File  Edit  View  Tools  Help          [Menu]

[Open] [Save] [Export]    [Tool Icons]



         [Canvas - Image/Video Display]
         [with polygon/line overlays]



[Timeline Scrubber]  (for videos)
```

```
Annotations:          Properties:
 region 1            Name: [region 1    ]
 region 2            Type: Polygon
 line 1              Vertices: 2
```

7.2 Mouse Interactions

Action	Result
Click (polygon mode)	Add vertex
Double-click	Close polygon
Escape key	Finish polyline (leave open)
Click on vertex	Select vertex
Drag vertex	Move vertex
Ctrl + Click on vertex	Delete vertex
Click on shape	Select shape
Delete key (shape selected)	Delete shape

7.3 Visual Styling (MVP)

- Polygons/lines: Solid color (e.g., cyan or yellow for visibility)
- Selected shape: Different color or thicker line
- Active vertex: Highlighted (e.g., larger circle)
- Line width: 2-3 pixels for visibility
- Vertex markers: Small circles (5-7 pixel radius)

Custom colors and line styles are deferred to post-MVP.

8 MVP Scope Definition

8.1 In Scope for MVP

☒ Image file loading and display

☒ Video file loading with frame selection via scrubber

☒ Polygon creation (click to add vertices, double-click to close)

- ☒ Polyline creation (Escape to finish without closing)
- ☒ Vertex editing (drag to move)
- ☒ Shape selection and deletion (Delete key)
- ☒ Individual vertex deletion (Ctrl+Click)
- ☒ Default sequential naming
- ☒ Name editing via properties panel
- ☒ Export to YAML and JSON with normalized coordinates
- ☒ Reload annotation file with media for editing
- ☒ Native file dialogs
- ☒ Basic menu and toolbar

8.2 Out of Scope for MVP

- ☐ Undo/redo functionality (nice-to-have)
- ☐ Video playback with ROI overlay (nice-to-have)
- ☐ Zoom and pan controls (nice-to-have)
- ☐ Custom colors and line styles
- ☐ Different region/line types beyond naming
- ☐ Batch processing multiple files
- ☐ Integration with CV processing pipelines
- ☐ Advanced shape operations (merge, split, etc.)
- ☐ Image preprocessing or enhancement tools

9 Future Enhancements

9.1 Post-MVP Features

9.1.1 Phase 2

- Undo/redo support
- Zoom and pan controls
- Video playback with overlay visualization
- Keyboard shortcuts for all operations
- Copy/paste/duplicate shapes

9.1.2 Phase 3

- Custom colors and styles per annotation
- Annotation grouping and layers
- Templates and presets
- Grid and snap-to-grid
- Measurement tools (distance, area)

9.1.3 Phase 4

- Batch annotation mode
- Export to additional formats (CSV, XML)
- Integration APIs for CV pipelines
- Plugin system for extensibility
- Collaborative editing features

10 Development Milestones

10.1 Milestone 1: Basic Infrastructure (Week 1-2)

- Set up project structure and development environment

- Implement main window and basic UI layout
- Implement image loading and display
- Implement canvas widget for drawing

10.2 Milestone 2: Annotation Creation (Week 3-4)

- Implement polygon creation with mouse clicks
- Implement line/polyline creation
- Implement visual rendering of shapes
- Implement shape selection

10.3 Milestone 3: Editing Capabilities (Week 5-6)

- Implement vertex dragging
- Implement shape and vertex deletion
- Implement naming and properties panel
- Implement annotation list view

10.4 Milestone 4: Video Support (Week 7-8)

- Implement video frame extraction
- Implement timeline scrubber control
- Integrate frame selection with canvas

10.5 Milestone 5: Data Persistence (Week 9-10)

- Implement YAML export
- Implement JSON export
- Implement annotation file loading
- Implement media+annotation reload workflow

10.6 Milestone 6: Polish and Testing (Week 11-12)

- Cross-platform testing (macOS, Linux)
- Bug fixes and refinements
- Documentation
- Release preparation

11 Testing Strategy

11.1 Unit Tests

- Coordinate normalization and denormalization
- Data serialization/deserialization (YAML/JSON)
- Geometry utilities
- Annotation data model

11.2 Integration Tests

- File I/O operations
- Media loading (images and videos)
- Export/import round-trip

11.3 Manual Testing

- UI interactions across platforms
- Edge cases (very large/small images, long videos)
- File format compatibility
- User workflow validation

12 Documentation Requirements

- User guide (org format)
- Installation instructions per platform
- Code documentation (docstrings)
- Example annotation files
- Architecture decision records (ADR) for key technical choices

13 Risk Assessment

Risk	Likelihood	Impact	Mitigation
Qt licensing complexity	Low	High	Use PySide6 (LGPL) if needed
Video codec support issues	Medium	Medium	Use OpenCV's codec support, document
Cross-platform UI differences	Low	Medium	Test early on all target platforms
Performance with large videos	Medium	Medium	Optimize frame extraction, add progress
Coordinate precision issues	Low	High	Use double precision, add validation

14 Success Criteria

The MVP will be considered successful when:

1. A user can load an image or video file
2. A user can create and edit polygons and lines
3. A user can export annotations to YAML/JSON with normalized coordinates
4. A user can reload and edit a previous annotation session
5. The application runs stably on macOS and Linux
6. The exported data is compatible with downstream CV systems
7. The user experience is intuitive and responsive

15 Appendix

15.1 Dependencies (Initial)

```
PyQt6>=6.6.0
opencv-python>=4.8.0
Pillow>=10.0.0
PyYAML>=6.0
numpy>=1.24.0
```

15.2 Reference Materials

- Qt for Python documentation: `https://doc.qt.io/qtforpython/`
- OpenCV Python tutorials: `https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html`
- PyYAML documentation: `https://pyyaml.org/wiki/PyYAMLDocumentation`

15.3 Glossary

- **ROI**: Region Of Interest - a polygon defining an area to monitor
- **Counting Line**: A line segment across which objects are counted
- **Normalized Coordinates**: Coordinates scaled to [0.0, 1.0] range
- **Polyline**: A series of connected line segments (open polygon)
- **Vertex**: A point defining a corner of a polygon or endpoint of a line
- **Annotation**: A generic term for any polygon or line defined by the user

16 Notes

This document serves as the primary reference for the ROIDS application development. It should be updated as requirements evolve or technical decisions are made.

The document prioritizes simplicity and functionality for the MVP while maintaining a clear vision for future enhancements.