OFIQ C# Wrapper Library Requirements

Project Overview

Create a .NET 8 wrapper library for the Open Source Face Image Quality (OFIQ) C++ library, providing a modern, type-safe, and performant C# API for facial image quality assessment.

Target Framework

- .NET 8.0 (Long Term Support)
- Multi-platform: win-x64, linux-x64, linux-arm64, osx-x64, osx-arm64
- Language: C# 12 with modern features

Core Requirements

1. Native Interoperability

- Use P/Invoke to interface with OFIQ native library (libofiq_lib.so/libofiq_lib.dylib/ofiq_lib.dll)
- Handle memory management and resource cleanup
- Provide safe abstractions over unsafe native calls
- Support both Release and Debug builds of native library

2. API Design

- Main Entry Point: OFIQEngine class implementing IDisposable
- Configuration: OFIQConfiguration for JAXN configuration management
- **Results**: FaceImageQualityAssessment with typed results
- Quality Measures: Enumeration of all 28 ISO/IEC 29794-5 measures
- Error Handling: Structured exception hierarchy

3. Data Structures

- Image Support multiple image formats (PNG, JPEG, BMP)
- LandmarkPoint and Landmarks Facial landmark data
- QualityMeasureResult Native score and quality value
- FaceImageQualityAssessment Complete assessment results
- PreprocessingResults Access to intermediate processing data

4. Quality Measures Support

Wrap all 28 quality measures defined in ISO/IEC 29794-5:

- 0x41: UnifiedQualityScore
- 0x42: BackgroundUniformity
- 0x43: IlluminationUniformity
- 0x44-0x45: Luminance (mean and variance)
- 0x46: UnderExposurePrevention

- 0x47: OverExposurePrevention
- 0x48: DynamicRange
- 0x49: Sharpness
- 0x4A: NoCompressionArtifacts
- 0x4B: NaturalColour
- 0x4C: SingleFacePresent
- 0x4D: EyesOpen
- 0x4E: MouthClosed
- 0x4F: EyesVisible
- 0x50: MouthOcclusionPrevention
- 0x51: FaceOcclusionPrevention
- 0x52: InterEyeDistance
- 0x53: HeadSize
- 0x54-0x57: CropOfTheFaceImage (left, right, above, below)
- 0x58-0x5A: HeadPose (yaw, pitch, roll)
- 0x5B: ExpressionNeutrality
- 0x5C: NoHeadCoverings

5. Configuration System

- Load JAXN configuration files
- Support relative and absolute model paths
- Quality mapping configuration via sigmoid functions
- Default configuration fallback

6. Image Processing

- Support common image formats via System.Drawing.Common
- Memory-efficient image loading
- Cross-platform image format support
- EXIF metadata handling

7. Performance Requirements

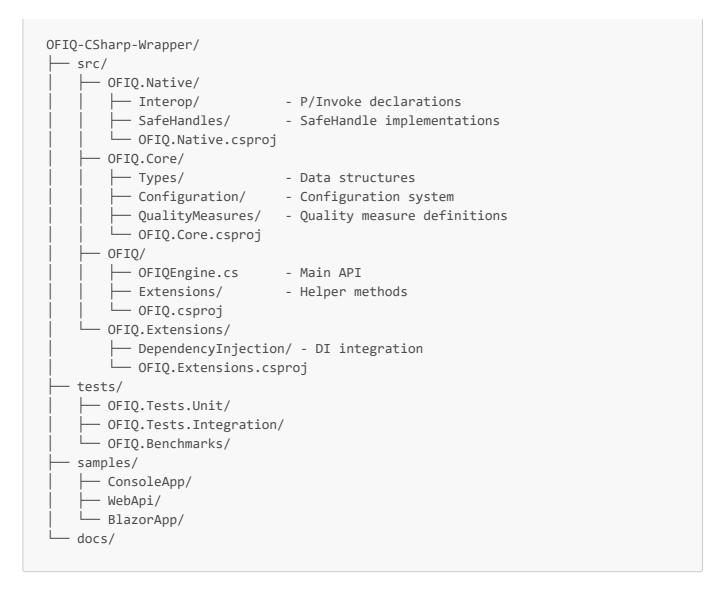
- Minimal overhead over native calls
- Efficient memory usage for image processing
- Thread-safe operations where applicable
- Support for batch processing

8. Error Handling

- Structured exception types (OFIQException, ConfigurationException, ImageLoadException)
- Native error code mapping to .NET exceptions
- Resource cleanup on errors

Technical Implementation

Project Structure



Dependencies

- System.Drawing.Common Cross-platform image processing
- System.Text.Json Configuration serialization
- Microsoft.Extensions.DependencyInjection Optional DI support
- **xUnit** Testing framework
- BenchmarkDotNet Performance testing

Build and Deployment

- Multi-target for all supported platforms
- Native library bundling strategy
- NuGet package creation
- Native AOT compilation support
- CI/CD pipeline configuration

Testing Requirements

Unit Tests

- All public API methods
- Configuration loading and validation

- Error condition handling
- Memory management verification

Integration Tests

- End-to-end quality assessment
- Cross-platform compatibility
- · Performance benchmarks
- · Memory leak detection

Sample Applications

- Console application for command-line usage
- Web API for RESTful interface
- Blazor application for web interface
- Performance benchmarking tool

Documentation

- XML documentation for all public APIs
- README with quick start guide
- API reference documentation
- Sample code and tutorials
- Performance guidelines

Compliance

- MIT license compatibility
- Proper attribution of OFIQ dependencies
- Cross-platform compatibility testing
- Security best practices for native interop

Success Criteria

- 1. Full API coverage of OFIQ functionality
- 2. Cross-platform compatibility
- 3. Performance within 5% of native calls
- 4. Comprehensive test coverage (>90%)
- 5. Production-ready deployment packages
- 6. Complete documentation and samples

Future Enhancements

- Async/await support for long-running operations
- GPU acceleration integration
- Custom quality measure extensions
- Real-time video processing
- Cloud deployment optimizations