LYYN SW plugin specs

Background

Outdoor and underwater video imaging is often plagued by low visibility due to weather, low-light or imperfect illumination. Low visibility is typically a cause of disappointment for the purchaser of a surveillance system. Additionally, the effective use of surveillance video, for incident detection and response, is often degraded by low visibility.

When we look at something, signals from red-sensitive, blue-sensitive and green-sensitive receptors in the eye retina travel to centers in the brain, where we interpret them, understand them and recognize patterns or objects. Equal signal levels in the different receptors means that we perceive a gray color. If there is the slightest hint of unbalance in the red, green and blue signals, the eye and brain can differentiate between many thousands of color shades and intensities. Thus color, if at all existent in a picture, is a powerful characteristic for object identification and extraction from a scene.

This is why the LYYN process is better than a pure contrast enhancer: it helps the human brain use its strengths, i.e. color separation and object identification even in an apparently "gray" scene.

What it does

The algorithm utilizes normal color video. Each video frame is processed in real-time and the color and luminance of each pixel is modified to increase overall visibility. Even marginal differences in color and contrast can be used to enhance object visibility. The result is video images that constantly self-adjust so that an observer can focus on surveillance objectives.

Requirements

The following requirements are for a generic CPU-architecture. FPGA requirements are more implementation specific.

Color model

RGB color. The algorithm can utilize other color models but this adds 2 color conversions per pixel.

Memory

A standard C-implementation can work in-place. We do recommend that memory for 3 frames is available, this includes space for supporting data structures.

Operating system, language, frameworks

No specific operating system required.

Direct access to image data, preferably planar representation.

C-language preferred.

The algorithm can be implemented with common image-processing frameworks.

Floating point operations are nice but not strictly required.

Processing

If we assume a typical BGRA or RGBA 32-bit pixel representation and approximately 6 operations, add or multiply, per pixel per frame we get the following example rates

Frames per second	Frame width	Frame height	Pixels per second	Operations per second
25	640	480	7 680 000	46 080 000
30	640	480	9 216 000	55 296 000
30	1920	1080	62 208 000	373 248 000
60	1920	1080	124 416 000	746 496 000

Example & existing solutions

FPGA

LYYNs hardware offerings utilize code running on FPGAs. IP-blocks for Xilinx Artix 7 and older Actel FPGAs are available.

CPU

Generic C-implementation.

C implementations for iOS.

CPU-GPU

Objective-C and OpenGL ES as well as Swift with Metal for iOS.

Typical integration project steps

Pre-study

- Algorithm presentation & walkthrough
- Presentation of customer technical environment
- Requirements documentation
- Implementation project planning including work package definitions

Implementation

- Customer implementation

or

- LYYN implementation

For more information please contact info@lyyn.com