Eclipse Identity Recognition Corporation
1154 60th Street
Brooklyn, NY 11219

FusionIR

Road Map

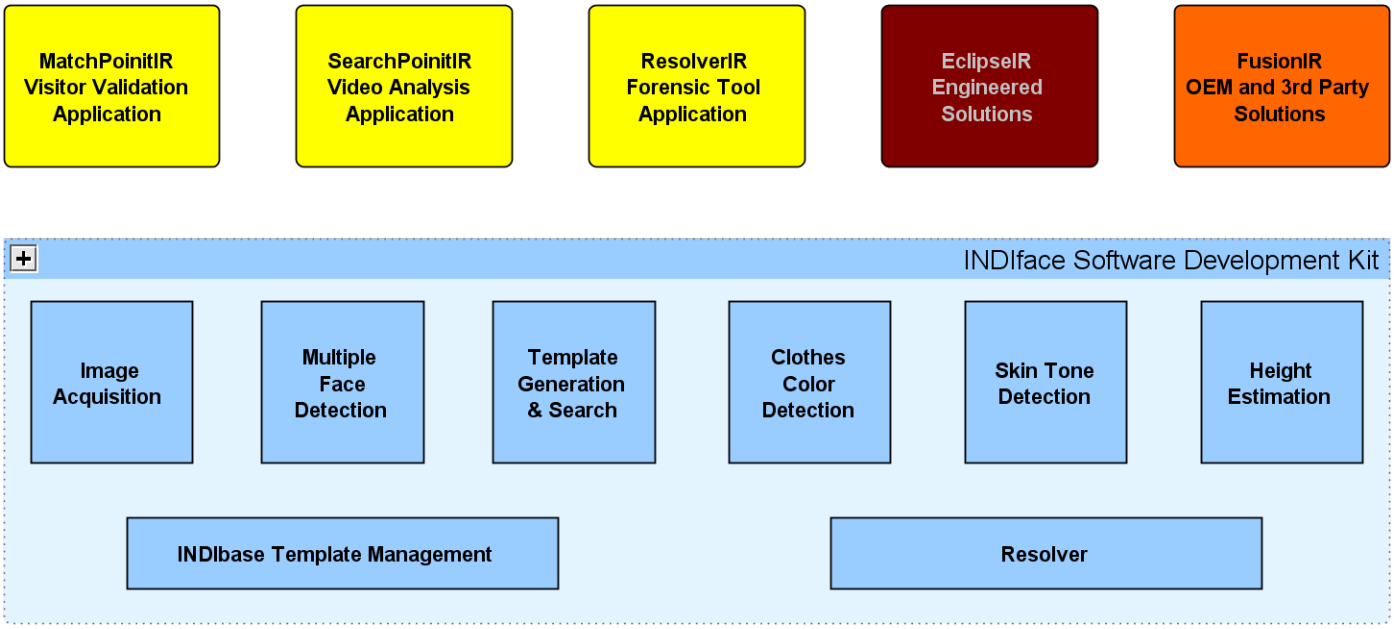
March 7, 2013

Version 1.10

Eclipse Identity Recognition Corporation™ provides Personal Identity Recognition via its face-based analytics known as INDIface™. It is used to detect faces and facial features and then generate face templates for searching and matching. In addition, the INDIface analytics can also determine skin tone, upper and lower clothes color, and an estimated height of a person. The INDIbase system provides for the enrollment, storage, management, retrieval, and matching of INDIface templates. INDIbase will also manage facial and other images as well as personal identification and characteristic data.

FusionIR™ packages EclipseIR's INDIface technologies into a software developer kit (SDK) to allow an original equipment manufacturer (OEM) to include the technologies in their existing systems or a third party developer to create custom applications with INDIface. EclipseIR's own applications (recapped in the table below) use the same SDK that is available to OEM customers.

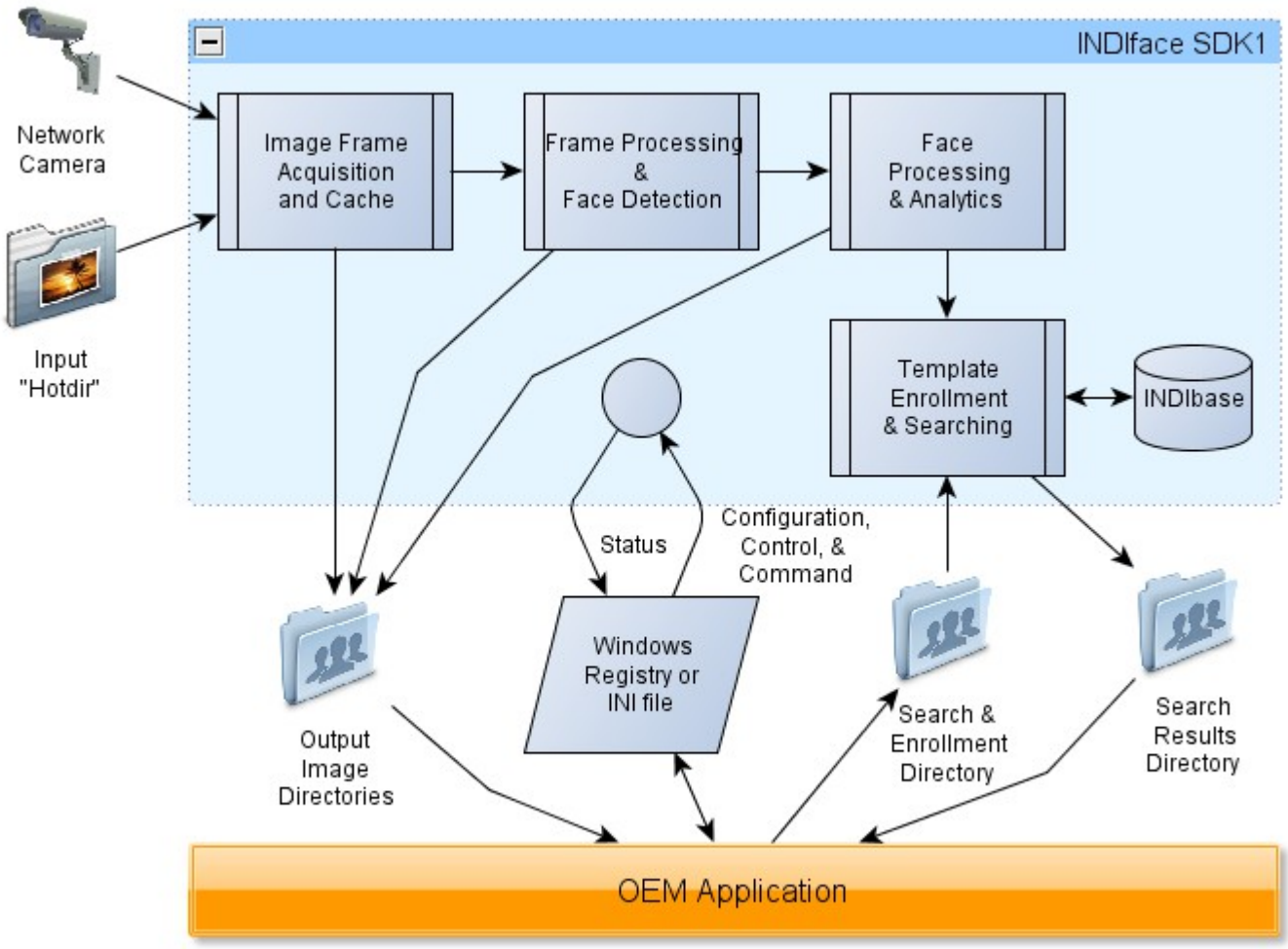
Match*Point*IR™	Search*Point*IR™	ResolverIR™
Monitor entrances and access points in commercial and public facilities	**Search and monitor live and stored video for analysis and investigation**	**A video forensics tool to locate individuals based on biometric characteristics**
Detects faces at an entrance or "choke-point," and identifies individuals by matching detected faces to a person of interest database. Alarms can be set to notify of known troublemakers, intruders, and VIPs.	Searches stored video files to detect and locate specific persons of interest. Ideal in video investigations for locating lost persons, troublemakers, or a known predator.	Incorporating multi-modal techniques, Resolver utilizes advanced biometric video analytics to search for persons of interest based not only the facial recognition but also on skin color, height, upper & lower body clothing color.



First Generation

The initial INDIface technology is based upon a mature core and has been deployed since 2009 Q4. Additional enrollment management functions were added in 2010 Q1 and face-based analytics were integrated into SDK1 in 2010 Q2. Further an authentication mode has been added in 2011 Q4 and enrollment with manual eye locations has been added 2012 Q3.

The first generation INDIface software development kit (SDK1) is primarily for Windows-based stand-alone systems with up to 50,000 enrollments. In the short term, SDK1 has been built and tested on 64-bit Ubuntu Linux using INI style files for configuration, control, and command. It is expected that it could be deployed on on most Linux- or BSD-based platforms. At the moment the binary implementation of SDK1 is a Windows console application (IfSearch.exe) and a few third-party DLLs.



Performance Expectations

On a typical PC and typical environment for digitized NTSC images, multiple facial detection takes 200 to 250 milliseconds. For each potential face, another 150 to 200 milliseconds is consumed. Then to sort each template against 25,000 enrolled faces takes another 100 milliseconds. So, the bottom line is that the typical image is typically processed in less than a second. *YMMV*.

Second Generation

While SDK1 is essentially single-threaded, the second generation of INDIface (If^2) will focus on multiple threads to split the work across multicore, multiprocessor servers, and even, multiple-server (i.e. blade) systems.

If^2 will introduce an NoSQL-based INDIbase enrollment repository and a plug-in Frame Source Architecture and an Image Manager Service to accept video channels from a wider variety of sources.

Transformational versus Reactive Systems

The basic building blocks of the If^2 technologies (as described below) are considered parts of a transformational system. In transformational systems, the results of a process are *not* dependent upon anything but the input data. That is, the same input frame image with the same parameters will (should?) always produce exactly the same list of potential faces and so on.

The only interaction between the basic processes is a prerequisite relationship: Face Processing depends upon Frame Processing; Analytic Processing and Template Processing depend upon Face Processing; and Results Fusion depend upon Face Processing, Analytic Processing, and Template Processing.

This certainty allows the mathematic processing and test data to be developed once and then compiled, optimized, and unit tested on different processor-operating system environments.

The basic transformational INDIface technologies can be exposed to the outside through reactive systems. Reactive systems respond to external events or messages and their protocols are often implemented as state machines. Different physical interfaces and protocols can be quickly, reliably, and predicatively implemented around the basic If^2 processes. *Once the building blocks are solid, the plumbing is quick and easy.*

Two implementations (described below) are built into the If^2 plans, but others such as BioAPI, SOAP, etc. as well as those to support individual OEMs can be quickly implemented.

Parallelism and Scalability

Microcomputer processors are not achieving ever faster clock speeds. In order to scale up to handle more work in a reasonable amount of time, we have to take advantage of parallelism across multiple threads, processor cores, processors, and blades. The following quotes from Network World illustrate:

> *Parallel computing is the primary way that processor manufacturers are dealing with the physical limits of transistor-based processor technology. Multiple processors—or cores—are joined together on a single integrated circuit to provide increased performance and better energy efficiency than using a single processor. Multicore technology is now standard in desktop and laptop computers. Mobile computing devices like smartphones and tablets are also incorporating multicore processors into their designs.*
>
> *The problem with multicore computing is that software applications no longer automatically benefit from improvements in processor performance the way they did in the past. Those benefits can only be realized by writing applications that expect and take advantage of parallelism.*[1]

Processing a frame through INDIface is a little like a snake digesting a rodent; rodents come in different shapes and sizes and some are more tender than others. And, the more snakes you have, the faster the rodent population is going to decline. In the second generation technology, we will look to do a little more math to improve the accuracy of the results and rely on parallelism to continue to achieve sub-second results on the typical image. That parallelism can be applied on three levels:

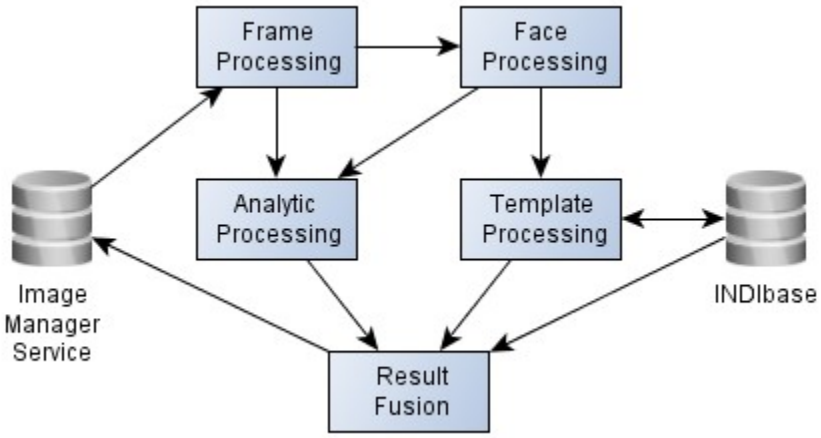
- Splitting up pixels or vectors across multiple threads
- Splitting up functional blocks across multiple cores
- Splitting up the work across multiple servers.

Also, the data will be moving through NoSQL servers which provide easier scalablity than tradtional databases.

1 Introduction to The State of Parallel Programming: The Parallel Programming Landscape, a white paper by *Dr. Dobb's* sponsored by Intel.

Basic Building Blocks

The basic operations of If^2 can be broken into five functional blocks in addition to the separate processes for INDIbase and Image Manager Service SQL interfaces: Frame Processing, Face Processing, Analytic Processing, Template Processing, and Result Fusion.



Frame Processing receives frames from the Image Manager Service transforms it and then pre-processes it (in parallel) by skin detection, variance detection, and luminance stretching. Regions of interest are then processed (in parallel) for frontal-face object detection.

Face Processing involves detecting face features (in parallel), determining which set of face feature locations is optimum (in parallel), normalizing the face image based upon the best feature locations, and calculating one or more vectors (in parallel) for the normalized face.

Analytic Processing currently has three components which can be performed in parallel for each qualified face: (1) Extracting characteristic face color pixels from the normalized face image and calculating the face's skin color. (2) Extracting characteristic color pixels below the detected face in the input frame and determining the clothes color of upper and lower body. (3) Based upon the feature locations, estimate the height of the person belonging to the face.

Template Processing includes sorting face vectors by similarity of match to enrolled vectors and management of the enrollment of persons and their face vectors (collected into templates).

Results Fusion is the final destination for each input frame. The results from all of the other processes are collected, evaluated, and distributed to the appropriate places.

Implementations

In the second generation, FusionIR will be wrapped in two architectures. The second generation software development kit (SDK2) will mimic the interface of SDK1. An application program interface (API2) will be based upon messages passed via TCP/IP sockets; the command, control, and configuration packets can be between applications on a single system or across a wide area network. Both SDK2 and API2 can be implemented as Windows services and 'BSD/Linux daemons. The API2 will serve as the basis for interfacing with other standard APIs, such as BioAPI and the Onvif SOAP standard for video analytics.

Performance Expectations

The actual amount of math needed per frame will vary even more in If^2 because some frames will allow fewer pixels to be evaluated than others. But we do know some starting places for calculations: frontal face finding will still be about one second per megapixel and template generation will still be about five faces per second.

Sorting templates by face appearance is the most straightforward function to estimate processing needs and ability to scale. Multiply the number of layers in each vector (typically 200 to 400) by the number of search vectors (one to a few) by the number of enrolled vectors (few to millions) and divide by the available mega-flops.

From there, we can estimate what a blade will be able to do. Let's assume a mythical blade with sixteen 2GHz cores and 8GB of RAM. Further assume that one core will keep the OS happy for networking, memory management, etc. and another core will be marshaling with the other blades to keep the processes moving, leaving fourteen cores to do the actual math. A blade could handle input from about fifty cameras. A blade could process about seventy faces. A blade could handle analytics for probably 150 faces. A blade could handle over half a million enrolled vectors.

Third Generation and Beyond

While the second generation of INDIface will concentrate on building upon and improving the current capabilities in a more sustainable and predictable form, the third generation will introduce even more capabilities to INDIface, such as vectors that are tailored to the face's apparent gender, age, and race (by facial structure, not skin tone).

The following chart indicates what capabilities will be available as INDIface progresses:

		SDK1	SDK2	API2	ThirdGen
Platform		"Standard PC"	"Server Class"	Multiple Server	Multiple Server
Operating Systems					
	Windows 32-bit	√	√	TBD	TBD
	Windows 64-bit		√	√	√
	'BSD/Linux 32-bit		TBD	TBD	TBD
	'BSD/Linux 64-bit	•[2]	√	√	√
C^3 Method (Configuration, Control, Command)					
	Windows Registry	√	√		
	INI file	•			
	XML file		√		√
	IP Socket			√	√
	Multiple Sources		√		√
Process Threading		Single	Multiple	Multiple Machine	Multiple Machine
INDIbase					
	File-based	√	√		
	NoSQL-based		√	√	√
	Maximum Enrollment	50,000	750,000	Multi Million	Multi Million
Pre-Processing					
	Transformation (scale, aspect, rotate)	√	√	√	√
	Crop		√	√	√
	Skin Detection		√	√	√
	Variance Detection		√	√	√
	Luminance Stretching		√	√	√
	Named Regions		√	√	√
	Dual Resolution		√	√	√
Multiple Face Detection		√	√	√	√

2 Tested on Ubuntu 10.10 with plans to test on FreeBSD, Centos, and TurboLinux

		SDK1	SDK2	API2	ThirdGen
Feature Location					
	Eyes	√	√	√	√
	Nose & Mouth		√	√	√
	Multiple Feature Detection		√	√	√
	Eye Location Consistency	√			
	Face Feature Congruence		√	√	√
Template Generation		√	√	√	√
	Lighting Correction		√	√	√
	Multiple Vector				√
	Aspect Correction		√	√	√
Enrollment					
	Person Mode (Create & Remove)	√	√	√	√
	Add & Delete Faces	√	√	√	√
	Remove All	√	√	√	√
	Batch Enrollment		√	√	√
	Re-enroll		√	√	√
	Non-face Enrollment (Bad Face, Distinguishing Marks, Associated Images)		√	√	√
	Specify Eye Locations	√	√	√	√
	Associate Tracking				√
	Pruning via Vector Averaging				√
Resolver					
	Detection Quality	√	√	√	√
	Eye Location Consistency	√			
	Face Feature Congruence		√	√	√
	Clothes Color	√	√	√	√
	Skin Tone	√	√	√	√
	Height	√	√	√	√
	Match Confidence		√	√	√
	Enrolled Characteristics		√	√	√
	Apparent Demographics (Age, Race, & Gender Estimation)	•	Preliminary	Preliminary	√