<https://pypi.org/project/harvesters-util/#about-harvester>

Harvester was created to be a public and friendly image acquisition library for all people who those want to learn computer/machine vision. Technically speaking, Harvester is a Python library which is responsible for the following tasks

Image acquisition

Device manipulation

Image data visualization (optional)

Harvester consumes image acquisition libraries, so-called GenTL Producers. If you have an officially certified GenTL Producer and GenICam compliant machine vision cameras, then Harvester supply you the acquired image data as numpy array to make your image processing task productive.

You can freely use, modify, distribute Harvester under Apache License-2.0 without worrying about the use of your software: personal, internal or commercial.

Currently, Harvester is being developed and maintained by the motivated contributors from all over the world.

Where is the name from?

Harvester’s name was coming from the great Flemish painter, Pieter Bruegel the Elder’s painting so-called “The Harvesters”. Harvesters harvest a crop every season that has been fully grown and the harvested crop is passed to the consumers. On the other hand, image acquisition libraries acquire images as their crop and the images are passed to the following processes. We found the similarity between them and decided to name our library Harvester.

* **The GenApi-Python Binding**: A Python module that communicates with the GenICam reference implementation.
* **A GenTL Producer**: A library that has C interface and offers consumers a way to communicate with cameras over physical transport layer dependent technology hiding the detail from the consumer.
* **The GenTL-Python Binding**: A Python module that communicates with GenTL Producers.
* **Harvester**: A Python module that consists of Harvester Core and Harvester GUI.
* **Harvester Core**: A part of Harvester that works as an image acquisition engine.
* **Harvester GUI**: A part of Harvester that works as a graphical user interface of Harvester Core.
* **A GenICam compliant device**: It’s typically a camera. Just involving the GenICam reference implementation, it offers consumers a way to dynamically configure/control the target devices.

Harvester Core

Harvester Core is an image acquisition engine. No GUI. You can use it as an image acquisition library which acquires images from GenTL Producers through the GenTL-Python Binding and control the target device (it’s typically a camera) through the GenApi-Python Binding.

Harvester Core works as a minimalistic front-end for image acquisition. Just importing it from your Python script, you should immediately be able to set images on your table.

You’ll be able to download the these language binding runtime libraries from the EMVA website, however, it’s not available as of May 2018, because they have not officially released yet. Fortunately they are in the final reviewing process so hopefully they’ll be release by the end of 2018.

If you don’t have to visualize acquired images at high frame rate, the combination of Harvester Core and Matplotlib might be ideal for that purpose.

Tasks Harvester Core does for you

The main features of Harvester Core are listed as follows:

Image acquisition over GenTL Producers

Multiple loading of GenTL Producers in a single Python script

GenICam feature node manipulation of the target device

Note that the second item implies you can involve multiple types of transport layers in your Python script. Each transport layer has own advantages and disadvantages and you should choose appropriate transport layers following your application’s requirement. You just need to acquire images for some purposes and the GenTL Producers deliver the images somehow. It truly is the great benefit of encapsulation by the GenTL Standard!

On the other hand, Harvester Core could be considered as a simplified version of the GenTL-Python Binding; actually Harvester Core hides it in its back and shows only intuitive interfaces to its clients. Harvester Core just offers you a relationship between you and a device. Nothing more. We say it again, just you and a device. If you need to manipulate more relevant GenTL modules or have to achieve something over a hardcore way, then you should directly work with the GenTL-Python Binding.

Pixel formats that Harvester Core supports

Currently Harvester Core supports the following pixel formats that are defined by the Pixel Format Naming Convention:

Mono8, Mono10, Mono12, Mono16, RGB8, RGBa8, BayerRG8, BayerGR8, BayerBG8, BayerGB8, BayerRG16, BayerGR16, BayerBG16, BayerGB16

Der Name Harvester stammt von dem großen flämischen Maler Pieter Bruegel dem Älteren, dessen Gemälde "Die Erntearbeiter" genannt wird. Die Erntemaschinen ernten in jeder Saison eine voll ausgewachsene Ernte, die an die Verbraucher weitergegeben wird. Auf der anderen Seite erwerben Bibliotheken zur Bilderfassung Bilder als ihre Ernte, und die Bilder werden an die nachfolgenden Prozesse weitergegeben. Wir fanden die Ähnlichkeit zwischen ihnen und beschlossen, unsere Bibliothek Harvester zu nennen.

Harvester ist eine Python-Bibliothek, die für folgende Aufgaben zuständig ist

Aufnahme von Bildern

Geräte-Manipulation

great benefit of encapsulation by the GenTL Standard!

Ein GenTL-Produzent ist ein Softwaretreiber, der die GenTL-Schnittstelle implementiert, um einer

eine Anwendung oder eine Softwarebibliothek in die Lage versetzt, auf Hardware zuzugreifen und diese auf generische Weise zu konfigurieren und

Bilddaten von einem Gerät zu streamen.

Ein GenTL Consumer ist jede Software, die einen oder mehrere GenTL Producer über die

definierte GenTL-Schnittstelle nutzen kann. Dies kann zum Beispiel eine Anwendung oder eine Softwarebibliothek sein.

GenICam

Harvester verwendet Bilderfassungsbibliotheken, sogenannte GenTL-Producer. Wenn Sie einen offiziell zertifizierten GenTL-Producer und GenICam-konforme Bildverarbeitungskameras haben, dann liefert Ihnen Harvester die erfassten Bilddaten als Numpy-Array, um Ihre Bildverarbeitungsaufgabe produktiv zu machen.

Das Ziel des GenICam GenTL-Standards ist es, einen generischen Weg zur Aufzählung von Geräten zu bieten

einem System bekannte Geräte aufzulisten, mit einem oder mehreren Geräten zu kommunizieren und, wenn möglich, Daten von

dem Gerät an den Host zu übertragen, unabhängig von der zugrunde liegenden Transporttechnologie. Dies ermöglicht einer

Software von Drittanbietern die Verwendung verschiedener Technologien zur Steuerung von Kameras und zur Erfassung von Daten in einer

Transportschicht-unabhängig zu erfassen.

Der Kern des GenICam GenTL-Standards ist die Definition einer generischen Transport Layer

Schnittstelle (TLI). Diese Softwareschnittstelle zwischen der Transporttechnologie und einer Drittanbietersoftware

Software wird durch eine C-Schnittstelle zusammen mit einem definierten Verhalten und einer Reihe von standardisierten

Funktionsnamen und deren Bedeutung. Für den Zugriff auf diese Funktionen wird das GenICam GenApi-Modul

verwendet.

Das GenICam GenApi-Modul definiert ein XML-Beschreibungsdateiformat, um zu beschreiben, wie man

auf Gerätefunktionen zugreifen und diese steuern kann. Die Standard-Features-Namenskonvention definiert das

Verhalten dieser Funktionen.

Die GenTL Softwareschnittstelle deckt keine gerätespezifischen Funktionen des entfernten Geräts

Gerätes mit Ausnahme derjenigen zum Aufbau der Kommunikation. Der GenTL bietet einen Port, der den Zugriff

auf die Funktionen des entfernten Geräts über das GenApi-Modul.

Dies macht die GenTL zur generischen Software-Schnittstelle für die Kommunikation mit Geräten und

Daten von ihnen zu übertragen. Die Kombination von GenApi und GenTL bietet eine vollständige Software

Architektur für den Zugriff auf Geräte, z. B. Kameras.

Die Motivation hinter GenICam

Vor Jahren, als sich die Industriekameras weiterentwickelten und immer mehr Modelle, Schnittstellen und On-Board-Funktionen anboten, waren die Entwickler von Bildverarbeitungssystemen zunehmend frustriert über die mühsame Kamerasteuerung über Registerzugriffe und die Notwendigkeit, das Rad für fast jedes Projekt neu zu erfinden. Daher arbeiteten führende Hersteller von industriellen Bildverarbeitungskameras und Software zusammen, um einen effizienteren, branchenweiten Ansatz zu entwickeln. Im Jahr 2006 wurde der erste GenICam-Standard mit großem Erfolg veröffentlicht: Endlich war es möglich, Bildverarbeitungsanwendungen viel schneller zu entwickeln und den Quellcode für andere Projekte mit unterschiedlichen Kameras und Sensoren wiederzuverwenden und sogar eine API für verschiedene Kameraschnittstellen zu verwenden.

Selbstbeschreibende Kameras

Anstelle der meist unstandardisierten und mühsamen Kamerasteuerung über Register

Zugriff, setzt GenICam auf selbstbeschreibende Kameras und das GenAPI-Modul: Eine

XML-Datei mit standardisierter Syntax, die in der Regel in der Kamera gespeichert ist, kann

kann von jeder GenICam-konformen Software ausgelesen und zur Kamerasteuerung verwendet werden

ausgelesen und zur Kamerasteuerung verwendet werden - unabhängig von Hersteller, Schnittstelle und Funktionsumfang der Kamera. Darüber hinaus,

Eigenschaften wie Beschreibungen, Tooltips oder Einheiten an eine GUI weitergegeben werden.

Standardisierte Funktionen für die Wiederverwendung von Code

Innerhalb des GenICam-Standards definiert das SFNC (Standard Features Naming Convention)

Modul sowohl Feature-Namen als auch Feature-Eigenschaften definiert, so dass der Code

Code leicht mit anderen Kameras wiederverwendet werden kann - auch wenn sich nicht nur der Funktionsumfang

sondern auch, wenn sie mit einer anderen Schnittstelle oder von einem anderen

Hersteller stammen. Ein Beispiel,

nur die (obligatorische) Eigenschaft

Name AcquisitionStart (nicht

AcqStart) ist erlaubt. Für

herstellerspezifische Merkmale,

erlaubt die SFNC genügend

Flexibilität.

Unterstützte Kamera-Schnittstellen

GigE Vision, USB3 Vision, CoaXPress und der neueste Camera Link-Standard implizieren

GenICam-Konformität. Darüber hinaus kann GenICam-Konformität auch für andere Schnittstellen erreicht werden

andere Schnittstellen erreicht werden: Einige Hersteller bieten zum Beispiel die GenICam-Steuerung ihrer

ihre FireWire-Kameras an. Die ersten GenICam-kompatiblen MIPI CSI-2-Kameras werden

bald von Allied Vision auf den Markt kommen.

GenICam-APIs

Kamerahersteller, die den GenICam-Standard einhalten, bieten in der Regel ein

kostenloses SDK mit APIs für C und C++ sowie eine Viewer-Anwendung zur

Plug & Play-Kamerasteuerung. Natürlich können Sie auch Bibliotheken wie OpenCV integrieren.

Kommunikation mit Software von Drittanbietern

Während die GenAPI die Steuerung der Kamerafunktionen durch den Benutzer ermöglicht, übernimmt die GenTL

(Generic Transport Layer) die physikalischen Aspekte (z. B. ermöglicht sie

Auflistung verfügbarer Kameras) und sorgt für den Transport der Bilddaten von der

Kamera (dem GenTL-Produzenten) zum Host (dem GenTL-Konsumenten). Jede GenICam

Kamera kann mit Tools wie MATLAB verwendet werden, wenn diese eine GenTL-Schnittstelle zu

GenTL-Produzenten bieten. Sie müssen lediglich die Transportschicht und den Treiber der Kamera installieren.

GenICam und die Zukunft der eingebetteten Bildverarbeitung

Derzeit sind die meisten etablierten industriellen Bildverarbeitungsanwendungen wie Qualitätskontrolle

Qualitätskontrolle, industrielle Inspektion oder Verkehrskontrolle verwenden GenICam und laufen auf PCs. Mit

der zunehmenden Leistungsfähigkeit von Embedded Systemen werden viele dieser Anwendungen

werden viele dieser Anwendungen in den nächsten Jahren auf eingebettete Systeme umgestellt werden. Natürlich werden Entwickler

und Kunden, die an die Vorteile von GenICam gewöhnt sind, wären nicht bereit

nicht bereit sein, einen Schritt zurück zur Kamerasteuerung über Registerzugriff oder zur Verwendung von APIs wie

V4L2, die für viele industrielle Anwendungen nicht geeignet sind. Außerdem gilt es als

dass Kameras leicht ausgetauscht werden können, um eine bessere Leistung zu erreichen

oder um Kosten zu sparen. Mit Hunderttausenden von verkauften GenICam-Kameras pro Jahr,

wird die Umstellung der industriellen Bildverarbeitungsanwendungen auf eingebettete Systeme auch die

auch die Migration von GenICam. Darüber hinaus halten GenICam-Kameras mit integrierter

Bildvorverarbeitungsfunktionen in die Embedded Vision ein.

Mehr Informationen über GenICam

Weitere Informationen über GenICam finden Sie bei der EMVA