Software Development for Embedded Vision with the Basler pylon 5 Camera Software Suite

Embedded technologies can cut manufacturing costs in Vision systems, but they typically also involve significantly higher software development costs compared with standard PC applications. The pylon 5 Camera Software Suite helps resolve this dilemma by lowering the cost threshold for embedded applications.

Contents
1. Embedded Systems for Computer Vision .................. 01
2. Manufacturing Savings through Embedded Architectures ................................................. 01
3. A View toward Development Costs .......................................................... 01
4. The Embedded Cost Threshold ..................................................................... 02
5. Software Development with pylon ......................................................... 02
6. Software Development with pylon for Embedded Systems ....................... 03
7. Summary ........................................................................................................ 03

1. Embedded Systems for Computer Vision

The algorithms for automated image processing have traditionally required powerful computer systems. Even just a few years ago, embedded systems were simply not an option for more challenging applications such as face detection. Advances in technology have now produced processors that can handle the most complex image processing tasks with minimal power consumption, even on mobile devices (smartphones, tablets, etc). The chips used in the embedded range feature a CPU core together with a GPU (Graphics Processing Unit) and additional peripherals (bus systems, interfaces such as USB, HDMI, Ethernet, etc) all on one chip. This kind of SoC (System on Chip) contains all the essential components of a computer system. This technological progress is particularly impressive for the GPU: driven by the requirements of power-hungry applications such as video and gaming, today’s graphic processors in devices such as the latest smartphones can handle tasks that even a few years ago needed a full-fledged PC graphics card. Given its architecture optimized for graphic calculations, a GPU is particularly well suited for simultaneous processing of many parallel tasks. This property makes the GPU an ideal “processing workhorse” for typical computer vision algorithms. This is different from a classic CPU, which is usually designed for working through sequential tasks not necessarily related to vision algorithms. Various programming interfaces (such as Nvidia’s CUDA or OpenCL) provide the software developer with a relatively easy option for shifting individual computational functions from the CPU to the GPU, further optimizing the performance of the hardware.

2. Manufacturing Savings through Embedded Architectures

Compared with a PC sub-system built from discrete components, a comparable SoC is significantly cheaper. The reduced power consumption and significant space savings also promote more affordable designs. There is also potential for further cost reductions through the use of additional embedded components. Basler’s dart camera family offers powerful but affordable board-level camera modules that can be ideally integrated into an embedded hardware setup via affordable and flexible connections (USB3 or BCON for LVDS). This makes it possible for computer vision systems that just a few years ago cost thousands of dollars to today sell for several hundred dollars (or less).

![Fig. 1: Comparison of a classic PC-based setup with an embedded vision solution.](image)

3. A View toward Development Costs

One major advantage for classic, desktop PC-based vision systems is their comparably simpler setup. In the most straightforward case, there’s no more to it than a USB cable connected to the USB port of a standard Windows PC via plug-and-play. The development of vision software, particularly when using a Basler camera, is relatively uncomplicated. The Basler pylon Camera Software Suite allows for fast development of application software using popular development environments (such Visual Studio) and programming languages (such as C, C++). One key point is that software development runs on the same system as the applications being developed, which simplifies testing and debugging significantly.
Application prototypes to detect technical risks early can easily be created on the development computer, with no additional hardware needed.

What does standard software development for embedded systems look like? For starters, a more complex operating system is usually involved - normally some version of Linux more or less tailored to the hardware requirements of the system. In many cases developers need to familiarize themselves with a variety of different programming interfaces for various drivers, camera technologies, etc. Beyond this, the development process is made more complicated by the fact that embedded systems are normally designed on development computers that are not the target system for the planned application. In most cases, the development of an embedded Linux target system requires a Linux development computer (desktop PC), upon which the programming code is written and compiled. The compiled code is then copied to the target system for execution. Special tools allow the development computer to remotely debug the application running on the target system. The development of application prototypes is also a complex and time-intensive task in terms of software development for precisely the same reasons, as well as the fact that prototype hardware is usually required.

Taken as a whole, this development process is significantly more complex, requires more specialized knowledge from the developer and also takes longer. For obvious reasons, this pushes development costs higher than for a straightforward Windows desktop PC setup.

When evaluating the profitability of a development product, the expected unit count of the product being developed as well as the associated manufacturing costs are weighed off against the anticipated development costs. For low unit counts, there may hence be almost no manufacturing cost benefit. The goal is thus to keep development costs as low as possible, which for this application would tend to favor a classic desktop setup with standardized interfaces. For higher unit counts, however, the greater development costs can in some cases be justified, as lower manufacturing costs allow for rapid amortization of the development costs. There’s thus no one-size-fits-all answer for the profitability line for embedded development. Yet it is clear that cutting development costs is a potential key for making embedded products with embedded applications possible even for lower unit counts – which establishes the possibility of further improving the competitiveness of products under development.

5. Software Development with pylon

Basler’s pylon Camera Software Suite represents an ideal SDK (Software Development Kit) for developing Computer Vision applications with Basler cameras. pylon features a series of easy-to-use, GUI-based tools that help even non-developers set up their Basler cameras, such as for evaluation purposes or for the first-time configuration of the unit. In service situations, pylon also offers helpful utilities for troubleshooting. Software developers will also appreciate its core tool, the pylon Viewer, as an aid for creating program code.

pylon Viewer not only captures images from the camera easily, its feature tree provides access to all camera parameters. If a specific camera feature is selected using this tree (such as GainAuto), then the Viewer will present the developer with all relevant information for that feature, including C++ codes that can be adopted into the developer’s code via copy and paste.

Fig. 2: The higher the unit count, the greater the impact of production costs on overall costs. For lower unit counts, the likelihood is higher that an embedded approach will eat up any production costs savings through increased development costs.

4. The Embedded Cost Threshold

Fig. 3: When a camera feature is selected (in this case GainAuto), pylon Viewer displays all information of relevance to the developer, including the C++ codes.
Beyond this, pylon also includes a powerful SDK with APIs (Application Programming Interface) for C, C++ and .NET languages such as C#. The design of the API puts an emphasis on flexibility and ease-of-use, ensuring that camera applications can be developed for Basler cameras after even a short acclimatization period. Programming examples for all relevant application cases, and comprehensive documentation help further flatten the learning curve.

6. Software Development with pylon for Embedded Systems

What really separates pylon from the competition, however, is the fact that the respective API always looks the same no matter which operating system (Windows, Mac OS X, Linux), camera interface technology (USB3 Vision, GigE Vision, BCON, etc.) or even underlying processor architecture (ARM or x86) is being used. For this reason, we call pylon the „Unified Camera SDK.“

Not even pylon can fully eliminate the difficulties in embedded development mentioned above. But the pylon’s Unified approach does mean that embedded development can also be performed in part on non-embedded systems by significantly less experienced developers.

One example illustrates this effectively: Company X wants to bring a mobile passenger counter onto the market as a handheld device. It should allow market researchers to count passengers while walking through a train car, with no other actions required of the passengers. Based on the clear size restrictions for the device, the final product will feature the dart BCON, a small board-level camera with a small embedded platform built around an ARM processor running Linux.

The initial steps, however, involve determining which camera is appropriate and evaluating its optical performance. The entire process is aided by the fact that a model identical to the dart BCON camera is available with a USB 3.0 interface. As such, that camera can be connected using plug-and-play to a standard Windows PC. pylon Viewer can then be used to test out the camera’s various parameters, including the pylon Viewer’s recording function for capturing individual images or video sequences for more intense subsequent analysis. This setup can now be used to create initial software prototypes, and to implement and test algorithms for passenger counting, even by developers who normally have no contact with embedded development. Any technical risks are made clear in the earlier phases.

At some point, of course, the development product must still be ported to the Linux/ARM target system. Thanks to pylon’s Unified approach, the code developed to that point on a standard Windows system can be in large part reused and requires only a re-compile for the new target system — presuming that care was given to the portability of the non-pylon portions of the software during development. Depending on which target system is selected, some degree of code adaption will be required. When migrating from USB3 Vision to BCON — as is the case in this example — the image capture interface for example must be redone for BCON. On the whole, however, pylon ensures that the embedded portion of the overall development process (i.e. the portion of the application that must be developed directly on an embedded platform) is kept as minor as possible, including the option to delegate individual development packets to colleagues with no significant embedded development experience. When development costs are cut in this way, the investment cost analysis ends up correspondingly more positive.

7. Summary

If you’re looking solely at manufacturing costs, embedded systems are significantly cheaper than standard applications. When it comes to development costs, however, the opposite is usually true, meaning that the embedded approach is usually justified only if a certain unit count is produced.

pylon makes it possible for key portions of an embedded development project to be handled in the same way as a standard development project. This lowers the complexity and duration of a development project and makes it significantly easier to create prototypes (including earlier identification of technical risks). As such, pylon helps lower that unit threshold for the embedded approach.
Author

Frank Karstens
Product Platform Manager

Frank Karstens is Product Platform Manager (Host Software) at Basler AG. Karstens, who holds a degree in engineering, started at Basler in 2005 in the Components division. In his current position he holds responsibility for the host software on machine vision cameras (pylon) and embedded software and future developments in those fields.

Contact

Frank Karstens – Product Platform Manager
Tel. +49 4102 463 492
Fax +49 4102 463 46492
E-mail: frank.karstens@baslerweb.com

Basler AG
An der Strusbek 60-62
22926 Ahrensburg
Germany

Basler AG
Basler is a leading manufacturer of high-quality digital cameras for applications in manufacturing, medicine, traffic and retail. Product development is led by the demands of industry. The cameras offer simple integration, compact sizes, excellent image quality, and an outstanding price/performance ratio. Basler has more than 25 years of experience in image processing. The company is home to more than 500 employees at its headquarters in Ahrensburg, Germany and its subsidiaries and sales offices in Europe, Asia and the USA.