UDC 004.424

Using of fast unmanaged code in C# language to accelerate the processing of images from unmanned aerial vehicle camera

Stepanov V. Y., Zuyonok A. V.
Belarusian National Technical University

It is known that different languages are based on completely different sets of basic concepts. Programming languages (like C#) use a term such as a managed context. This means that C# (more precisely CLR – Common Language Runtime) should work with fixed-size data types, because the code can be compiled by a JIT (Just-in-time) compiler under any of the supported target platforms (unless otherwise specified). There are a number of cases that govern the use of an unmanaged context, for example, in situations where the performance is critical (as in the case of processing images obtained from unmanned aerial cameras) where the standard GetPixel (...) method of the Bitmap class is proposed from the assembly System.Drawing (in System.Drawing.dll) to get the brightness of one pixel of the image.

The situation is complicated by the fact that the call of this method must be performed as many times as the number of pixels make our image. Taking into account the resolution of modern cameras, it is necessary to call the GetPixel (...) method more than one thousand times, which significantly reduces the overall system performance. In this case it is convenient to use a combination of managed code (for obtaining an image) and unmanaged code for organizing a quick access operation to an image through a pointers mechanism. This interaction is classified as an in-process and it is necessary to apply some operations that by default C# programmers are unavailable; since it is considered that the managed context is safer. For the interaction of managed with unmanaged code, in which unmanaged libraries are connected to the managed application, there is a Platform Invoke (p / Invoke) mechanism or in simple cases (for code) it is sufficient to use the unsafe keyword. For example:

```
Bitmap bSrc = (Bitmap)sourceImage.Clone();
BitmapData bmData = sourceImage.LockBits (new Rectangle (0, 0, sourceImage.Width, sourceImage.Height), ImageLockMode.ReadWrite, System.Drawing.Imaging.PixelFormat.Format24bppRgb);
unsafe
{
   byte* r = (byte*)(void*)bmData.Scan0;
   byte* g = (byte*)(void*)bmData.Scan0; g++;
   byte* b = (byte*)(void*)bmData.Scan0; b += 2;
}
```