
# GUIDE DE L'UTILISATEUR

Série BEAMAGE | Kit de développement (SDK)

121-105635

gentec-eo
PARTENARIAT de PRÉCISION

## GARANTIE

Les accessoires Gentec-EO pour diagnostics de faisceaux sont garantis contre tout vice de fabrication et de main-d'œuvre pour une durée d'un an à compter de la date d'expédition, lorsqu'ils sont utilisés dans des conditions de fonctionnement normales. La garantie ne couvre pas les dommages liés à une mauvaise utilisation ou à une pile qui fuit.

Gentec-EO Inc. réparera ou remplacera, à sa discrétion, tout accessoire pour diagnostics de faisceaux qui présente un défaut pendant la période de garantie, excepté dans le cas d'une mauvaise utilisation du produit ou une pile qui fuit.

La garantie est annulée si une personne non autorisée tente de modifier ou de réparer le produit.

Le fabricant ne peut être tenu responsable des dommages consécutifs, de quelque nature que ce soit.

En cas de mauvais fonctionnement, communiquez avec votre distributeur local Gentec-EO ou avec le bureau Gentec-EO Inc. le plus proche, afin d'obtenir un numéro d'autorisation de retour. Le matériel doit être retourné à :

Gentec Electro-Optics, Inc.
445, St-Jean-Baptiste, bureau 160
Québec, QC
Canada, G2E 5N7

Téléphone : (418) 651-8003
Télécopieur : (418) 651-1174
Courriel : service@gentec-eo.com

Site Web : http://www.gentec-eo.com

## RÉCLAMATIONS

Pour bénéficier d'un service sous garantie, communiquez avec votre représentant Gentec-EO le plus proche, ou envoyez le produit, accompagné d'une description du problème, avec l'assurance et le transport prépayés, au représentant Gentec-EO le plus proche. Gentec-EO Inc. n'assume aucune responsabilité en cas de dommage causé pendant le transport. Gentec-EO Inc. se réserve le droit de réparer ou de remplacer gratuitement le produit défectueux, ou de vous rembourser le prix d'achat. Toutefois, si Gentec-EO Inc. détermine que la défectuosité a été causée par une mauvaise utilisation, une modification, un accident ou des conditions de fonctionnement ou de manipulation anormales, celle-ci ne sera pas couverte par la garantie.

## Table des matières

LISTE DES FIGURES

# 1. Le kit de développement *Beamage SDK* de Gentec-EO

## 1.1. Qu'est-ce que le *Beamage SDK* ?

Le kit de développement *Beamage SDK* de Gentec-EO existe pour aider les clients à créer eux-mêmes leur propre interface utilisateur. Le kit de développement logiciel permet de faire des analyses personnalisées des images provenant de la caméra de profilométrie Beamage. Il permet aussi d'intégrer la caméra Beamage à un système sans utiliser le logiciel *PC-BEAMAGE* de Gentec-EO.

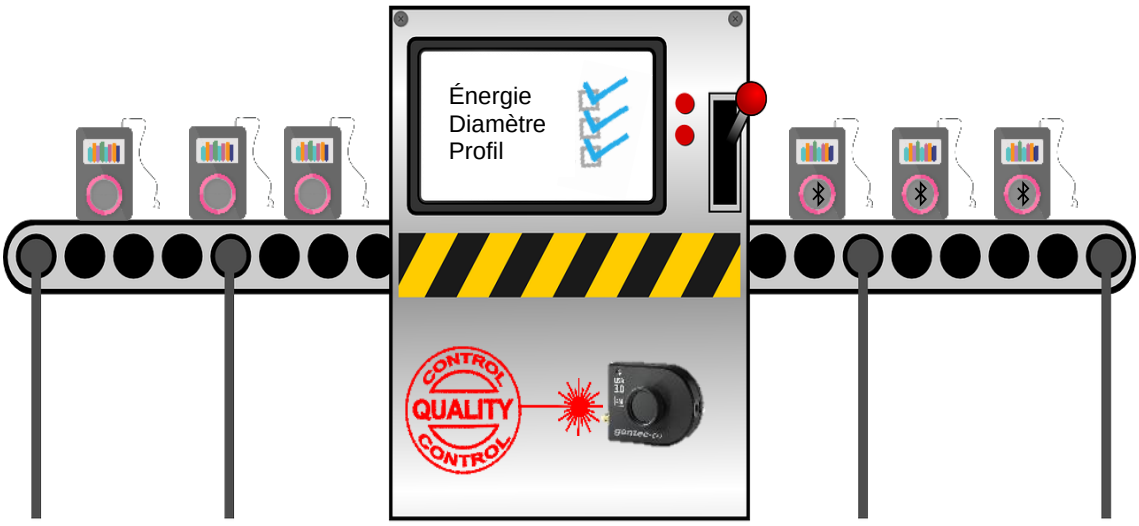



Figure 1 - Exemple conceptuel d'intégration du kit de développement *Beamage SDK* sur une ligne d'assemblage

Le kit de développement *Beamage SDK* est une *DLL* (bibliothèque de liens dynamiques ou *Dynamic Link Library*) qui communique avec les pilotes (*drivers*) de la caméra de profilométrie Beamage de Gentec-EO. La *DLL* inclut des fonctions pour contrôler la caméra et faire l'acquisition d'images, le tout dans le but de créer des applications *Windows*.

Cette *DLL,* écrite en C#, est compatible avec n'importe quel *langage .NET* : C#, Visual Basic, F#, etc. Nos clients peuvent ainsi créer des applications *Windows* personnalisées pour leurs besoins : analyse d'images, intégration, assurance qualité, sécurité et maintenance.

Voyez le schéma simple ci-dessous, qui illustre 2 différents moyens de gérer la caméra Beamage : le logiciel *PC-BEAMAGE* et le kit de développement *Beamage SDK*.

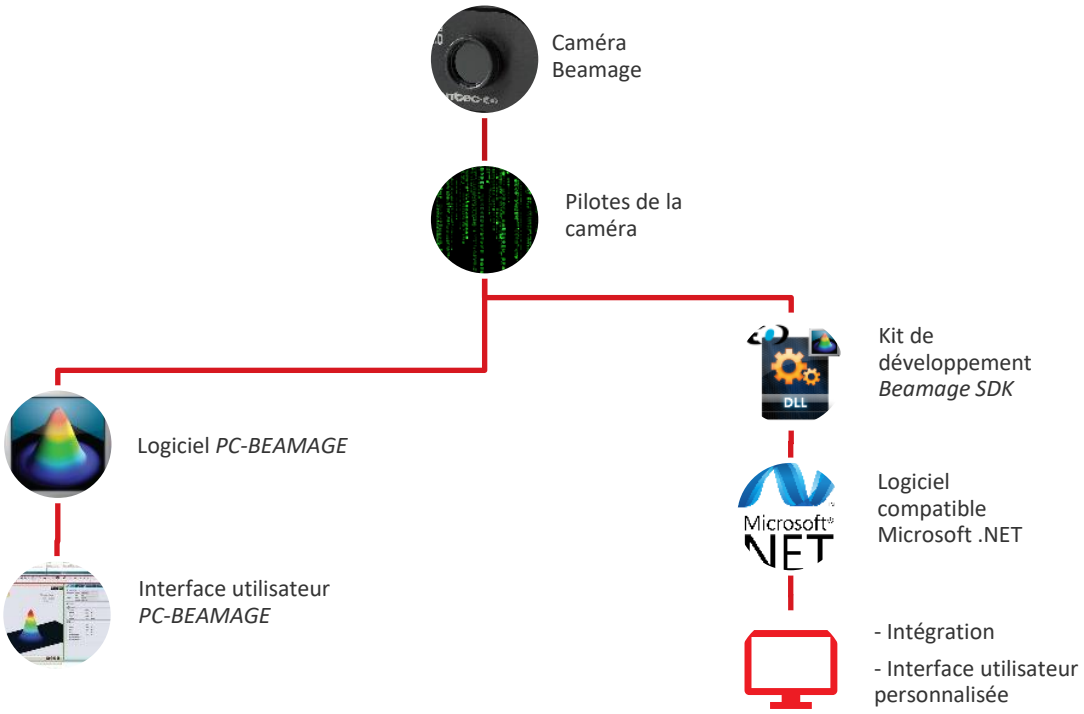



Figure 2 – Une nouvelle façon d'utiliser la caméra Beamage de *Gentec-EO* : le kit de développement *Beamage SDK* de *Gentec-EO*

### 1.2. L'option *pipeline .NET* de la caméra Beamage

Le kit de développement *Beamage SDK* de Gentec-EO offre des fonctions simples pour l'acquisition et l'analyse d'images, y compris le calcul de diamètre en 4 sigma XY. Pour bénéficier de toute la puissance du logiciel *PC-BEAMAGE* et intégrer ses fonctionnalités avancées à vos applications *Windows*, Gentec-EO met à votre disposition le *pipeline .NET*.

Plutôt que de communiquer directement avec la caméra comme le fait le *Beamage SDK*, le pipeline *.NET* communiquera avec le logiciel *PC-BEAMAGE* par l'entremise d'un pipeline RAM (mémoire vive ou *Random Access Memory*). Cela vous permet de créer ensuite votre application *Windows* personnalisée pour contrôler la caméra Beamage et bénéficier de la quasi-totalité des fonctionnalités du logiciel *PC-BEAMAGE*.

Voyez ci-dessous un schéma simple qui illustre le fonctionnement de l'option pipeline *.NET*.

Pour plus d'informations sur le pipeline *.NET*, référez-vous au manuel d'utilisation du logiciel *PC-Beamage*.

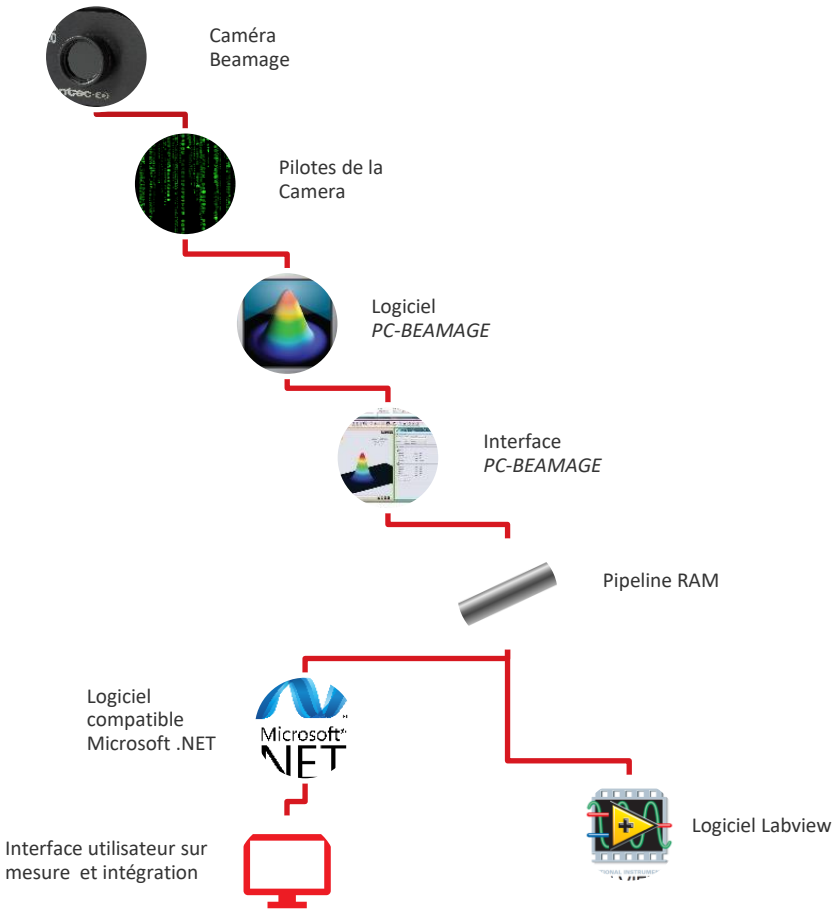



Figure 3 - Option de pipeline *.NET* pour la caméra Beamage

### 1.3. Comment démarrer avec le kit de développement *Beamage SDK* ?

Avant d'utiliser le *Beamage SDK*, veuillez-vous assurer d'être en mesure d'utiliser correctement la caméra *Beamage* et ses pilotes avec le logiciel *PC-BEAMAGE.* La dernière version du *PC-BEAMAGE*, les pilotes et les manuels de l'utilisateur peuvent être téléchargés à partir de l'onglet *Ressources* à la page Web suivante : https://www.gentec-eo.com/fr/diagnostic-de-faisceau-laser/profileurs-faisceaux.

Ce document contient toute la documentation de l'API ainsi que des exemples de code qui vous aideront à bien comprendre comment utiliser chacune des fonctions du kit de développement BSDK.

Gentec-EO met également à votre disposition une solution pour *Visual Studio*, le *Beamage SDK Samples*. C'est un exemple illustrant comment créer un objet de type caméra Beamage, s'y connecter, faire l'acquisition d'images et comment utiliser les nombreuses fonctions du kit de développement. La section Exemple de démarrage du présent document aide à mieux comprendre le *Beamage SDK Samples.*

### 1.4. Ce dont votre solution *Visual Studio* a besoin

Pour créer votre première application avec le kit de développement *Beamage SDK*, lancez *Visual Studio* et créez un nouveau projet.

Ensuite, téléchargez le fichier « *Gentec-EO Beamage SDK.zip* » depuis notre centre de téléchargement, et extrayez tous les fichiers. Trouvez le fichier *BeamageSDK.dll* et ajoutez-le comme référence à votre projet *Visual Studio*.

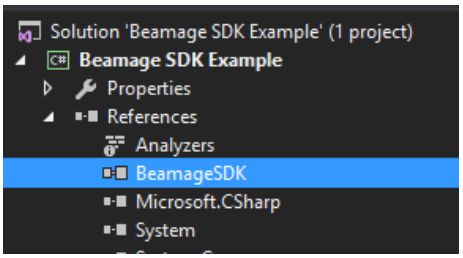



Figure 4 – Ajout de *BeamageSDK.dll* comme référence au projet *Visual Studio* en cours.

C'est tout ! Tout est maintenant en place pour commencer à programmer et à communiquer avec la caméra Beamage.

### 1.5. Débuter avec le code

Rien de mieux que des exemples de code pour comprendre comment utiliser le kit de développement *Beamage SDK*. Ici, nous pouvons voir un exemple de code qui instancie la classe BSDK et établit une connexion avec la première caméra Beamage que les pilotes trouveront sur l'ordinateur :

Exemple de code

```
using BeamageApi; //Importe la reference de l'espace de noms BeamageApi

namespace Beamage_SDK_Example
{
    public class BeamageSdkExample
    {
        BSDK bsdk; //Objet du Beamage SDK

        public BeamageSdkExample()
        {
            int index = 0; //Sélectionner la première camera trouvée

            bsdk = new BSDK(); //Instancie l'objet Beamage SDK

            bsdk.Detect();  //Cette méthode détecte et initialise toutes les caméras

            //Vérifier si au moins une caméra est trouvée
            if (bsdk.cameras.Count > 0)
            {
                //Afficher le numéro de série
                labelSerialNumber.Text =
                $"SerialNumber:
                { bsdk.cameras[index].Properties.GetSerialNumber()}";

               //Afficher le modèle
               labelBeamage.Text =
               String.Format("Camera: {0}",
               bsdk.cameras[index].Properties.Is4mSensor() ?
               "Beamage - 4M " : "Beamage - 3.0");
            }
        }
    }
}
```

C'est tout ! Une fois l'objet BSDK créé, toutes les autres fonctions du kit de développement *Beamage SDK* seront disponibles.

Pour plus d'exemples de code et pour voir comment utiliser toutes les fonctions, veuillez-vous référer à la section Exemple de démarrage.
Vous pouvez aussi consulter la Table des matières pour en apprendre davantage sur un aspect précis du kit de développement *Beamage SDK*.

### 1.6. Diagramme de classe du kit de développement *Beamage SDK*

Le kit de développement *Beamage SDK* est conçu pour être aussi simple que possible et être utilisé rapidement, sans maux de tête. Toutes les classes d'objets du kit de développement *Beamage SDK* commencent par la lettre B (pour Beamage).

La classe principale du kit de développement est un objet de classe BSDK. Les objets de cette classe contiennent des fonctions de base permettant d'obtenir le numéro de version du Beamage SDK, d'initialiser la caméra, de paramétrer la taille de la fenêtre d'affichage de l'image ainsi que de détecter et de se connecter à une caméra Beamage. La classe BSDK comprend les classes BCam (caméra) et BErrorsManager.

La classe BCam regroupe les fonctions de base pour contrôler la caméra. Elle contient les classes d'objets suivantes : BCamImg (image), BCamSettings (réglages et paramètres) et BCamProperties (propriétés en lecture seule).

La classe BErrorsManager aide le développeur à comprendre ce qui peut causer des erreurs lors de l'utilisation du kit de développement *Beamage SDK*.

Voici le diagramme de classe du kit de développement *Beamage SDK* :

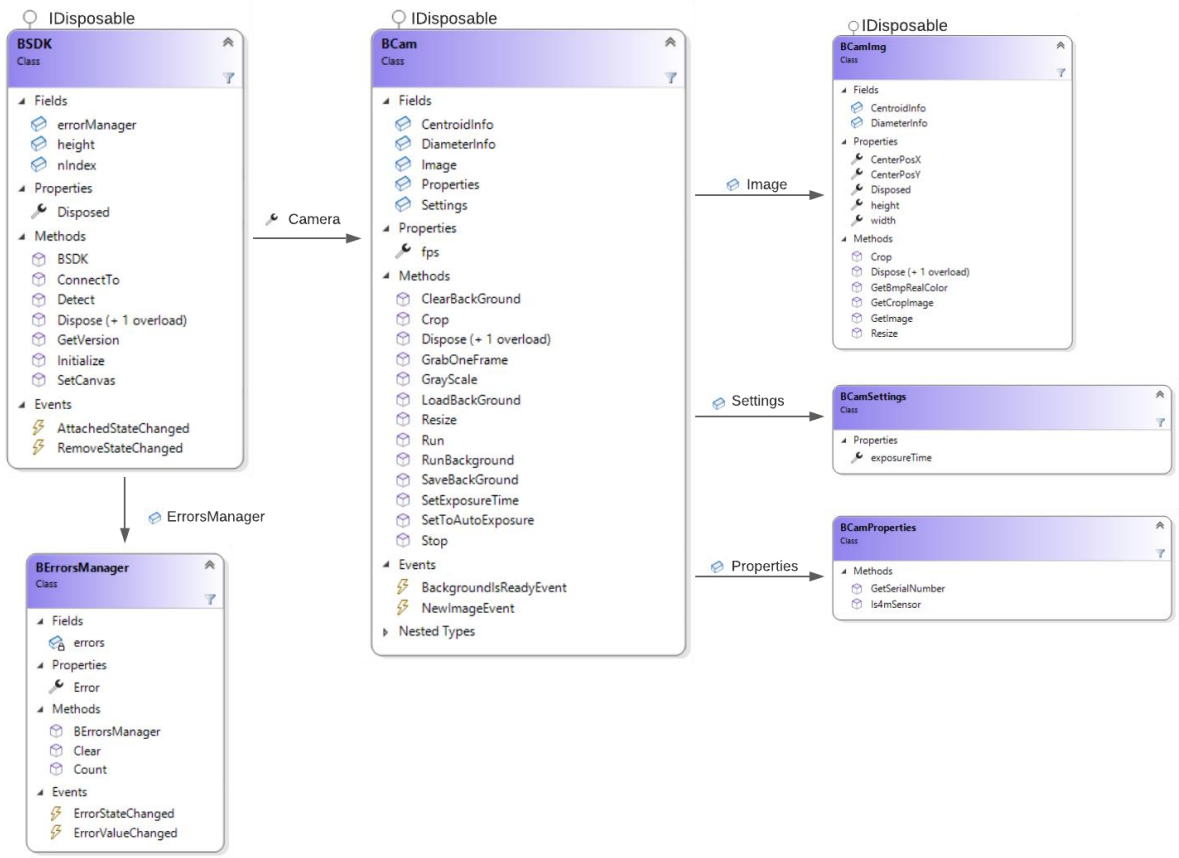



Figure 5 – Diagramme de classe du kit de développement *Beamage SDK*

# 2. Exemple de démarrage

Vous trouverez ci-dessous un exemple pour vous aider à démarrer avec le kit de développement *Beamage SDK*. L'exemple est écrit en langage C# et vous aidera à comprendre comment connecter une caméra et visualiser des images dans votre application.

## 2.1. Exemple C# avec visionneuse

### 2.1.1. Généralités

Cet exemple a été codé pour vous permettre de comprendre le fonctionnement du *Beamage SDK* et de le voir en action. Cette solution *Visual Studio*, disponible dans notre centre de téléchargement, inclut déjà comme référence le fichier *BeamageSDK.dll*. Cet exemple est une interface utilisateur très simple, élaborée à des fins de démonstration. Prenez note que comme le code fut écrit à des fins de démonstration uniquement, il n'est donc pas optimisé.

Voici deux captures d'écran du *Beamage SDK Exemple avec une caméra* :

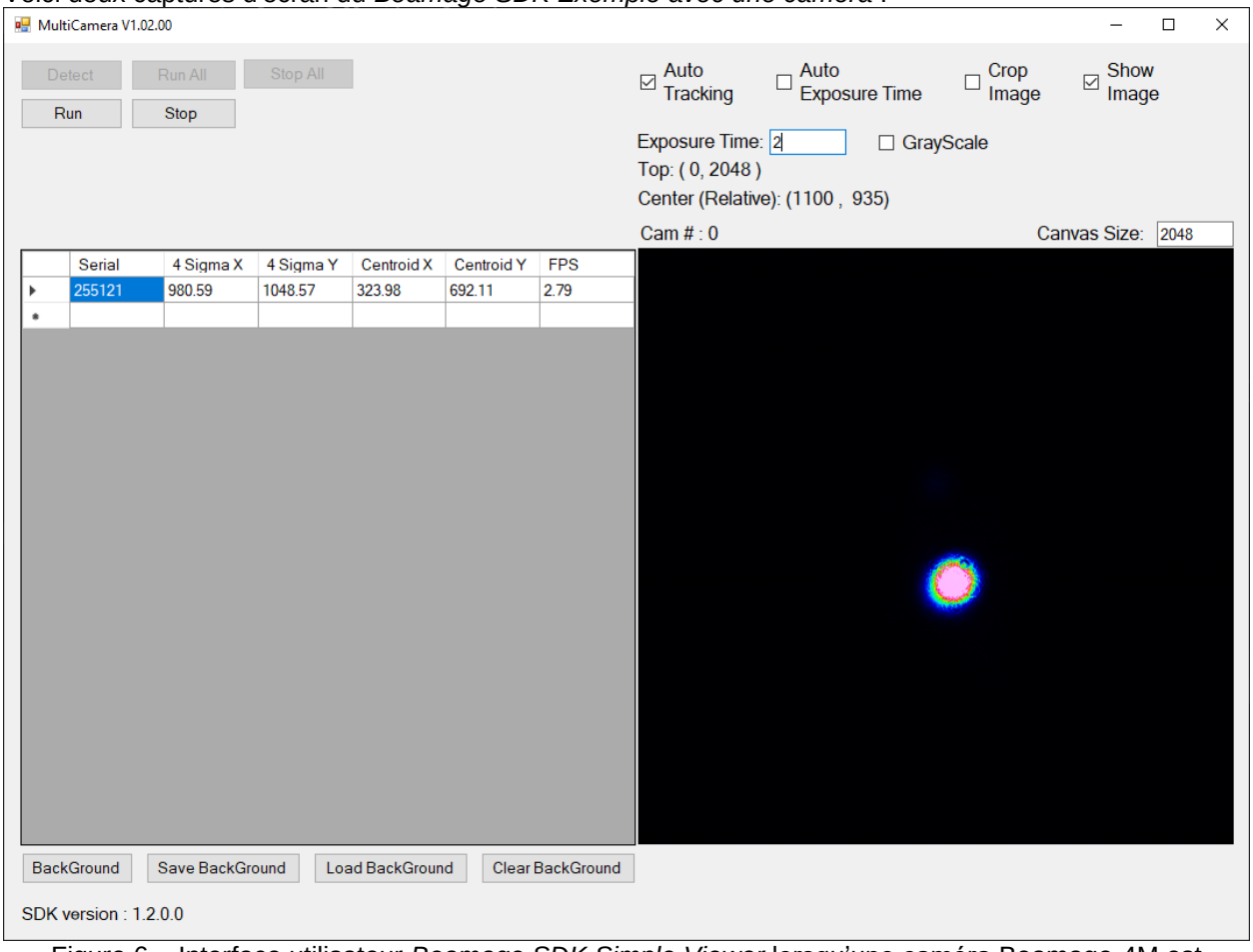



Figure 6 – Interface utilisateur *Beamage SDK Simple Viewer* lorsqu'une caméra Beamage-4M est connectée

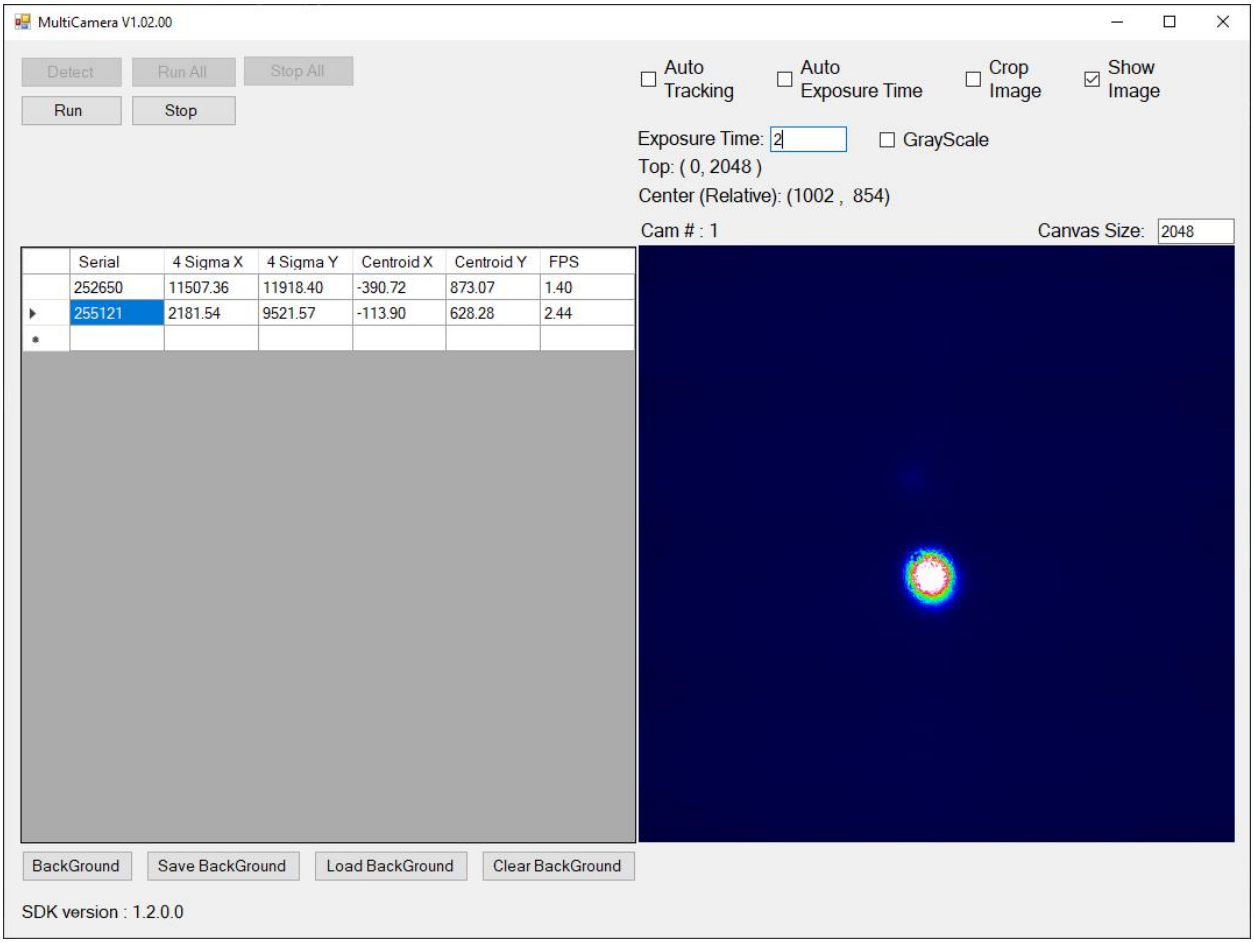



Figure 7 – Interface utilisateur *Beamage SDK Simple Viewer* lorsque deux caméras Beamage 4 M sont connectées

Le bouton **Detect** ouvre les pilotes de la caméra, détecte toutes les caméras Beamage branchées à l'ordinateur, puis initialise toutes les caméras détectées. Dans cet exemple, toutes les caméras trouvées afficheront leur numéro de série dans le sélecteur.

Voici le code source pour le bouton **Detect** :

Exemple de code

```
private void DectectCamerasButton_Click(object sender, EventArgs e)
{
  //Le SDK reconnaît toutes les cameras connectées.
  //Si aucune caméra n'est connectée, un message sera affiché.
  try
  {
    bsdk.SetCanvas(heigthROI);
    bsdk.Detect();
  }
  catch (Exception exception)
  {
    MessageBox.Show(exception.ToString());
  }

  cameras.Clear();
  dataGridViewCameras.Rows.Clear();

  foreach (var item in bsdk.cameras)
```

```
  {
    cameras.Add(item.Properties.GetSerialNumber());
    dataGridViewCameras.Rows.Add(item.Properties.GetSerialNumber(), "", "", "");
  }

  if (cameras.Count >= 1)
    SetDefautStateButtons(true);
  else
    MessageBox.Show("No camera has been detected", "Detection Cameras",
    MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
```

Après avoir appuyé sur le bouton **Detect**, les boutons **Run All**, **Stop All**, **Run**, **Stop** et **Background** seront rendus disponibles.

- **Run** : Démarre la capture d'images.
- **Stop** : Arrête la capture d'images.
- **Run All** : Démarre la capture d'images pour toutes les caméras.
- **Stop All** : Arrête la capture d'images pour toutes les caméras.
- **Auto Exposure Time** : Règle automatiquement le temps d'exposition de la caméra.
- **Exposure Time** : Règle manuellement le temps d'exposition lorsque **Auto Exposure Time** est désactivé. Les valeurs permises vont de 0,6 ms à 200 ms.
- **Auto Tracking** : Centre le faisceau sur le canevas.
  - Activez la fonction **Auto Exposure Time** avant d'utiliser cette fonction.
- **Crop Image** : Rogne la dernière image afin de la recadrer au centre.
- **Show Image** : Affiche la dernière image pour la caméra sélectionnée.
- **Canvas Size** : Règle la hauteur du canevas. La hauteur par défaut est de 2048.
- **Background** : Procède à une soustraction ISO de l'arrière-plan.
  - Cette fonction demandera à l'utilisateur de bloquer son laser. Appuyez sur le bouton **OK** pour commencer le processus.
  - Une fenêtre contextuelle (*pop-up*) apparaîtra lorsque le processus de soustraction de l'arrière-plan sera terminé.
  - La caméra Beamage doit en tout temps poursuivre la capture d'images pendant l'utilisation de cette fonction. Aucune validation du respect de cette condition n'est effectuée par voie logicielle. C'est la responsabilité de l'utilisateur de s'assurer que la caméra poursuit en tout temps la capture d'image.
- **Save Background :** Enregistre au format *.csv la soustraction de l'arrière-plan actuelle.
- **Load Background :** Charge une soustraction d'arrière-plan enregistrée.
- **Clear Background :** Efface la soustraction de l'arrière-plan actuelle.

### 2.1.2. Capture d'images

Pour montrer comment obtenir des images de la caméra, on a programmé un processus (thread) NewImageAllCam. Il s'agit d'un processus qui affichera en permanence la dernière image capturée par la caméra si la case **Show Image** est cochée. Cette tâche est lancée en appuyant sur le bouton **Run** et arrêtée en appuyant sur le bouton **Stop**.

Voici le code pour le bouton **Run** :

Exemple de code

```
private void ButtonCamRun_Click(object sender, EventArgs e)
{
    //Le SDK initialise la caméra sélectionnée et lance le fil NewImageAllCam.
    if (selectedIndex != -1 && !selectedIndexRunList.Contains(selectedIndex))
    {
       if (connectedCamList.Contains(selectedIndex))
       {
          bsdk.cameras[selectedIndex].Run();
          bsdk.cameras[selectedIndex].Resize(heigthROI);
          bsdk.cameras[selectedIndex].SetROI(topROI, heigthROI);
       }
       else
       {
          bsdk.cameras[selectedIndex].Connect();
          bsdk.cameras[selectedIndex].Run();
          bsdk.cameras[selectedIndex].Resize(heigthROI);
          bsdk.cameras[selectedIndex].SetROI(topROI, heigthROI);
          connectedCamList.Add(selectedIndex);
       }

       bsdk.cameras[selectedIndex].NewImageEvent +=
       new EventHandler(NewImageAllCam);

       textBoxExposureTime.Text =
       bsdk.cameras[selectedIndex].Settings.exposureTime.ToString("0.000");

       selectedIndexRunList.Add(selectedIndex);
       SetStateButtonsAllCameras(false);
   }
}
```

Après avoir appuyé sur le bouton **Run**, qui emploie la fonction ButtonCamRun_Click, la caméra capturera les images en continu et la fonction ShowImage montrera ces images dans l'interface utilisateur.

Voici le code de la fonction NewImageAllCam :

```
private void NewImageAllCam(object sender, EventArgs e)
{
  Image.GetThumbnailImageAbort myCallback =
  new Image.GetThumbnailImageAbort(ThumbnailCallback);

  //Affiche les valeurs 4 sigma, centroïde et FSP.
  if (dataGridViewCameras.InvokeRequired)
  {
     dataGridViewCameras.Invoke(new MethodInvoker(delegate
     {
       int nIndex = 0;
       foreach (var item in cameras)
       {
         dataGridViewCameras.Rows[nIndex].Cells["Serial Number"].Value   =
         bsdk.cameras[nIndex].Properties.GetSerialNumber();
         dataGridViewCameras.Rows[nIndex].Cells["4 Sigma X"].Value       =
         bsdk.cameras[nIndex].Image.diameterInfo.diameter4SigmaX.ToString("0.00");
         dataGridViewCameras.Rows[nIndex].Cells["4 Sigma Y"].Value       =
         bsdk.cameras[nIndex].Image.diameterInfo.diameter4SigmaY.ToString("0.00");
         dataGridViewCameras.Rows[nIndex].Cells["Centroid X"].Value      =
         bsdk.cameras[nIndex].Image.centroidInfo.centroidXPos.ToString("0.00");
```

```
          dataGridViewCameras.Rows[nIndex].Cells["Centroid Y"].Value      =
          bsdk.cameras[nIndex].Image.centroidInfo.centroidYPos.ToString("0.00");
          dataGridViewCameras.Rows[nIndex].Cells["FPS"].Value             =
          bsdk.cameras[nIndex].fps.ToString("0.00");
          nIndex++;}}));

        //Affiche l'image en couleur.
        if ((selectedIndex >= 0) && (selectedIndex < bsdk.cameras.Count) &&
           pictureBox.InvokeRequired && bShowImage)
        {
           pictureBox.Invoke(new MethodInvoker(delegate
           {
             Image image = bsdk.cameras[selectedIndex].Image.GetBmpRealColor();
             pictureBox.Image =
             image.GetThumbnailImage(image.Width / 4, image.Height / 4, myCallback,
             IntPtr.Zero);}));
        }
        else
        {
           pictureBox.Image = null;
        }

        //Affiche la valeur Top.
        if (labelTop.InvokeRequired)
        {
          labelTop.Invoke(new MethodInvoker(delegate { labelTop.Text =
          $"Top: ( {topROI}, {heigthROI} )"; }));
        }

        //Affiche la valeur du Centre.
        if (labelSize.InvokeRequired)
        {
           labelSize.Invoke(new MethodInvoker(delegate { labelSize.Text
           = $"Center (Relative): ({bsdk.cameras[selectedIndex].Image.CenterPosX} ,
           { bsdk.cameras[selectedIndex].Image.CenterPosY})"; }));
        }

        //Paramètre la valeur du temps d'exposition.
        if (textBoxExposureTime.InvokeRequired && bExposureTime)
        {
           textBoxExposureTime.Invoke(new MethodInvoker(delegate {
           textBoxExposureTime.Text =
           bsdk.cameras[selectedIndex].Settings.exposureTime.ToString("0.000");
           }));
        }

        //Calcul et paramètre les valeurs de Autotraking.
        for (int i = 0; i < bsdk.cameras.Count; i++)
        {
           if (bAutotraking)
           {
             if (bsdk.cameras[i].Image.CenterPosY > heigthROI / 2) topROI +=
             Math.Abs(bsdk.cameras[i].Image.CenterPosY - heigthROI / 2) / 2;
           else
             topROI -= Math.Abs(bsdk.cameras[i].Image.CenterPosY - heigthROI / 2)
             / 2;

           //Valeurs obligatoires :
           if (topROI < 0) topROI = 0;
           if (topROI > (2048 - heigthROI)) topROI = (2048 - heigthROI)
```

```
                bsdk.cameras[i].SetROI(topROI, heigthROI);
            }
        }
    }
}
```

Nous pouvons constater ici que l'application de l'exemple *Beamage SDK* effectue simplement une mise-à-jour en continu de la `pictureBox` avec une image BMP utilisant les mêmes couleurs que le logiciel *PC-BEAMAGE*. Le SDK Beamage met aussi à jour les valeurs nouvellement calculées de : 4 sigma XY, centroïde XY et FSP.

### 2.1.3. Calculs sur la mémoire tampon de l'image (*image buffer*)

Comme nous avons pu le voir dans l'exemple précédant, lorsqu'on appuie sur le bouton **Run**, le logiciel assigne l'événement `bsdk.cameras[index].newImageEvent` à la fonction newImage. Nous utilisons cette fonction pour démontrer la manière d'effectuer des opérations mathématiques à l'aide du kit de développement *Beamage SDK*.

Voici le code de la fonction newImage :

Exemple de code

```
    private void newImage(object sender, EventArgs e)
        {
            int index = 0;
          // Une nouvelle image a été captée par la caméra.
          // La fonction GetImage retourne un tableau d'entiers (int)
          // représentant l'image.
          // Avec les propriétés « width » (largeur) et « height » (hauteur), l'image
          //peut être retrouvée.
          // Ici, une simple moyenne de l'intensité de tous les pixels sera affichée.
            var image  = bsdk.cameras[index].Image.GetImage();
            int width  = bsdk.cameras[index].Image.width;
            int height = bsdk.cameras[index].Image.height;

            double pixelSum = 0.0;

            for (int i = 0; i < height; i++)
            {
                for(int j = 0; j < width; j++)
                {
                    pixelSum += image[i * height + j];
                }
            }

            // Calcule une moyenne.
            pixelSum /= (width * height);
        }
```

Nous avons vu comment effectuer des analyses et des opérations mathématiques directement sur la mémoire tampon de l'image. Ici, nous avons effectué un simple calcul de la moyenne de tous les pixels à des fins de démonstration.

### 2.1.4. Événements (*Events*)

Le kit de développement *Beamage SDK* contient quelques événements (*Events*). Le *Beamage SDK Example* montre comment les utiliser. Tout d'abord, affectez un *EventHandler* à un *Event*. Cette fonction sera appelée lorsque l'événement se déclenchera. Vous trouverez ci-dessous un exemple de code pour deux événements qu'on a affecté à des fonctions ainsi que la définition de ces fonctions :

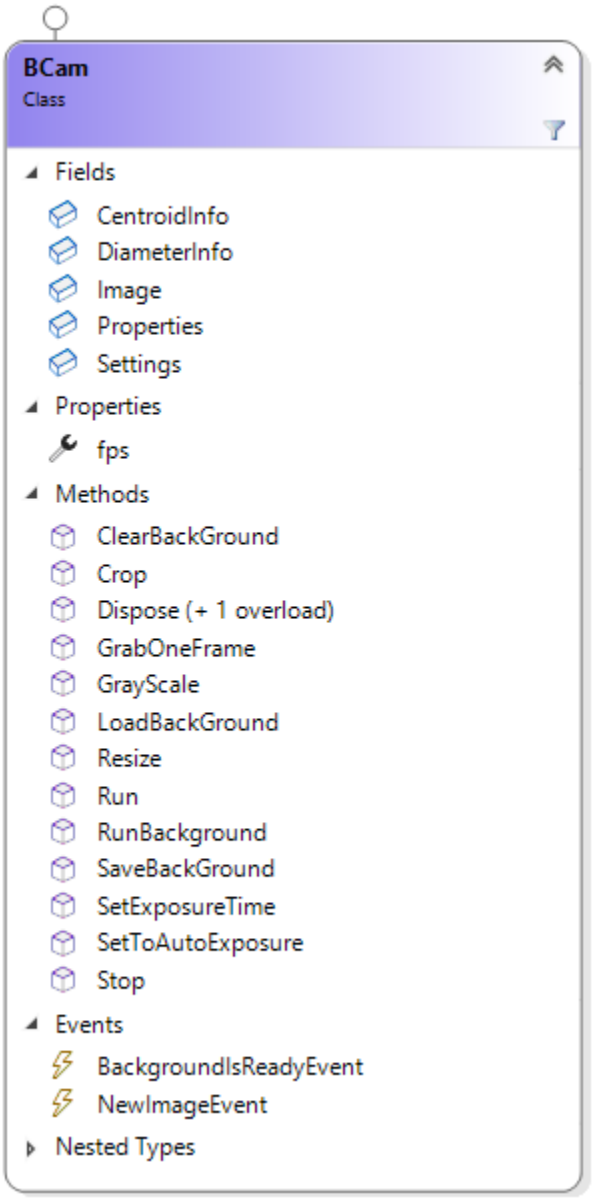
Exemple de code

```
// Assigne un événement pour l'ancrage et le retrait d'un périphérique.
bsdk.AttachedStateChanged += new EventHandler(attachedEvent);
bsdk.RemoveStateChanged += new EventHandler(removeEvent);

private void removeEvent(object sender, EventArgs e)
{
   // Une caméra a été déconnectée de l'ordinateur.
}

private void attachedEvent(object sender, EventArgs e)
{
   // Une caméra a été connectée à l'ordinateur.
}
```

# 3. Beamage API

Cette section donne une description de chaque classe et de chaque fonction du kit de développement *Beamage SDK*. Elle comprend aussi quelques exemples de code.

## 3.1. BCam




BCam est la classe du *Beamage SDK* donnant accès à la caméra. On peut, par exemple, accéder à la dernière image capturée en utilisant le champ BCamImg.

***Déclaration de classe (Class declaration)***

```
class BCam
```

***Attributs (Properties)***

```
BCamCentroid CentroidInfo;
```

Obtenir la valeur calculée actuelle du Centroïde XY.

```
BCamCentroid DiameterInfo;
```

Obtenir la valeur calculée actuelle du diamètre XY. Pour le moment, seul le diamètre 4 sigma est offert.

```
BCamImg Image;
```

Voir la section sur BCamImg.

```
BCamSettings camSettings;
```

Voir la section sur BCamSettings.

```
BCamProperties camProperties;
```

Voir la section sur BCamProperties.

```
float cameraFps { get; private set; };
```

Retourne le nombre actuel d'images prises par seconde par la caméra *Beamage*.

***Événements (Events)***

*event EventHandler NewImageEvent;*

Cet événement sera déclenché chaque fois qu'une nouvelle image sera capturée par la caméra *Beamage*.

Exemple de code

```
// Être avisé chaque fois qu'une nouvelle image est capturée
bsdk.camera.NewImageEvent += new EventHandler(newImage);

private void newImage(object sender, EventArgs e)
{
  // Une nouvelle image a été capturée par la caméra.
}
```

*event EventHandler BackgroundIsReadyEvent;*

Cet événement sera déclenché lorsque la soustraction de l'arrière-plan sera complétée.

Exemple de code

```
bsdk.camera.BackgroundIsReadyEvent += new EventHandler(backgroundIsReady);

private void backgroundIsReady(object sender, EventArgs e)
{
   MessageBox.Show("L'arrière-plan est prêt");
   bsdk.camera.BackgroundIsReadyEvent -= new EventHandler(backgroundIsReady);

}
```

***Fonctions (Functions)***

*void Crop(bool _crop)*

Attribuer à *_crop* la valeur *True* pour activer la fonction *Crop(int _CenterX, int _CenterY).* Attribuer à *_crop* la valeur *False* pour désactiver la fonction.

*void Dispose()*

La classe BCam implémente l'interface IDisposable et doit contenir une fonction Dispose.

*void GrabOneFrame()*

Capture une seule image et arrête ensuite la caméra.

*void GrayScale(bool _grayScale)*

Attribuer à *_grayScale* la valeur *True* pour changer l'image en niveaux de gris. Attribuer à *_grayScale* la valeur *False* pour désactiver.

*void RunBackground()*

Procède à une soustraction ISO de l'arrière-plan. Cette fonction doit acquérir un minimum de 10 images avant de pouvoir se terminer et c'est la responsabilité de l'utilisateur de s'assurer que 10 images ou plus sont acquises.

*void LoadBackGround(string _pathFileName)*

Attribuer à *_pathFileName* le chemin du fichier contenant la soustraction d'arrière-plan. La soustraction d'arrière-plan est enregistrée dans un fichier (*.csv). Voir la fonction *SaveBackGround(string _pathFileName)* pour plus d'information.

*void SaveBackGround(string _pathFileName)*

Attribuer à string *_pathFileName* le chemin du fichier où l'on souhaite enregistrer la soustraction de l'arrière-plan. Cette fonction enregistre l'information de la soustraction d'arrière-plan dans un fichier (*.csv).

*void ClearBackGround()*
Efface le tableau contenant la soustraction d'arrière-plan.

*void Resize(int _size)*
Attribuer à *_size* la hauteur désirée du canevas. La hauteur par défaut est de 2048. La largueur par défaut de 2048 n'est pas modifiable.

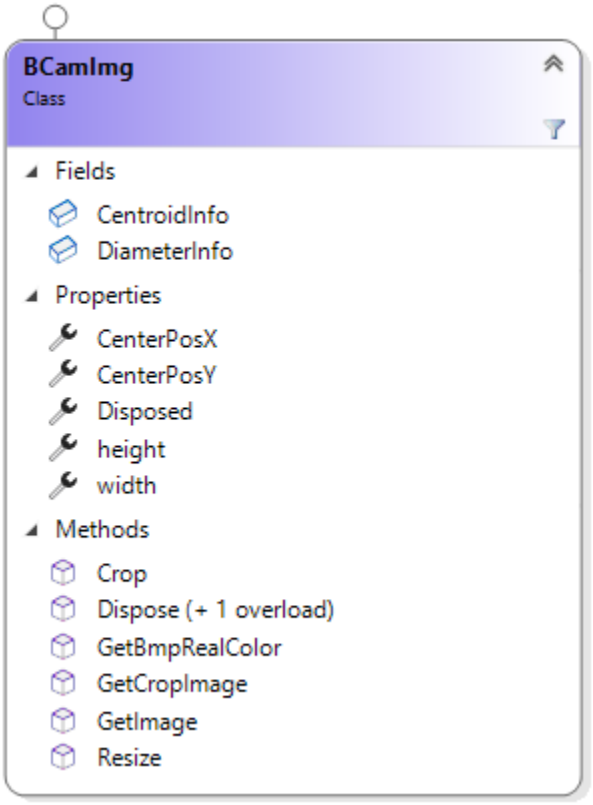
*void Run()*
Démarre la capture d'images par la caméra *Beamage* et continue de capturer des images jusqu'à ce que la fonction *StopRun()* soit appelée.

*void StopRun()*
Arrête la capture d'images de la caméra *Beamage* jusqu'à ce que la fonction *Run()* soit appelée.

*void SetExposureTime(float exposureTime)*
Règle manuellement le temps d'exposition de la caméra. Le temps d'exposition doit être compris entre 0,06 ms et 5 000 ms (5 secondes). Pour l'instant, aucune validation logicielle n'est effectuée. Le respect de ces limites est la responsabilité de l'utilisateur.

*void SetToAutoExposure(bool _autoExposure)*
Attribuer à *_autoExposure* la valeur *True* pour paramétrer la caméra en mode de temps d'exposition automatique.
Attribuer à *_autoExposure* la valeur *False* pour mettre la caméra en mode de temps d'exposition manuel et utiliser ensuite la fonction *SetExposureTime(float exposureTime)* pour modifier ce temps d'exposition.

### 3.2. BCamImg




BCamImg est une classe qui permet d'obtenir l'image brute dans une mémoire tampon ou en format BMP.

Le premier format est utile pour effectuer des analyses et des opérations mathématiques, tandis que le second est pratique pour afficher à l'écran.

***Déclaration de classe***
*class BCamImg*

***Attributs***
*BCamCentroid CentroidInfo;*
La valeur calculée actuelle du Centroïde XY.

*BCamCentroid DiameterInfo;*
La valeur calculée actuelle du diamètre XY. Pour le moment, seule la définition en 4 sigma est offerte.

*int CenterPosX*
La position en X du centre de la dernière image capturée. En lecture seule.

*int CenterPosY*
La position en Y du centre de la dernière image capturée. En lecture seule.

*int width*
La largeur de la dernière image capturée. En lecture seule.

*int height*
La hauteur de la dernière image capturée. En lecture seule.

***Événements***

La classe BCamImg ne contient pas d'évènements (Events).

***Fonctions***

*void Crop(int _CenterX, int _CenterY)*
Définir *_CenterX* et *_CenterY* pour recadrer l'image au centre. Cette fonction est utilisée pour améliorer les temps de calcul, car le SDK Beamage utilise moins de pixels autour du faisceau pour effectuer le calcul.

*Bitmap GetBmpRealColor()*
Cette fonction retourne une image en couleur comme celles du logiciel *PC-BEAMAGE*. La couleur correspond à l'énergie de chaque pixel tiré de l'image en tampon. Ci-dessous, l'échelle de couleurs utilisée :

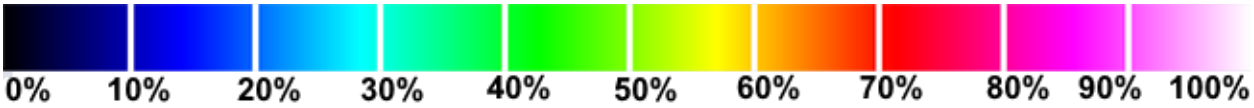


*Figure 8 – Échelle de couleurs*

*Short[] GetCropImage(int _CenterX, int _CenterY, int nWidth, int _Heigth)*
Définissez *_CenterX*, *_CenterY*, *nWidth* et *_Heigth* pour obtenir un tableau de *short* avec les données de la dernière image recadrée au centre.

*int[] GetImage()*
Retourne un tampon mémoire brut de la dernière image captée. Ce tampon mémoire est un tableau de données à une seule dimension. Le premier élément du tableau est le coin supérieur gauche de l'image, et les éléments suivants vont de gauche à droite, puis de haut en bas, se terminant en bas à droite de l'image. Cet ordonnancement des éléments du tableau est illustré par les flèches rouges ci-dessous.

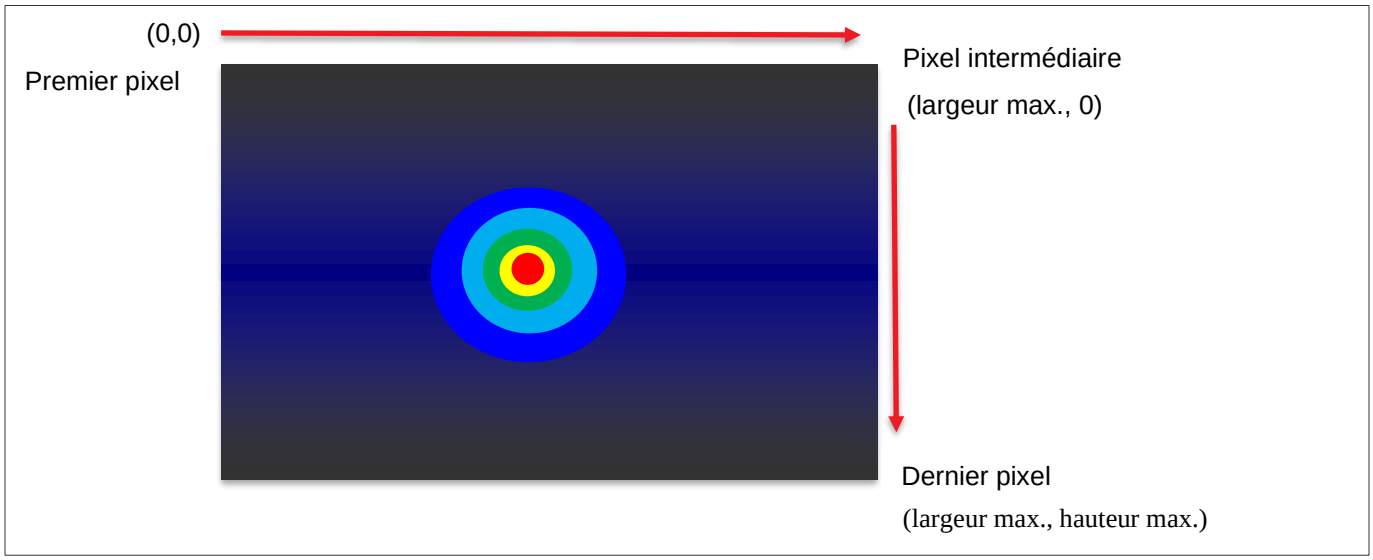



Figure 9 – Image obtenue par la caméra Beamage

Il est facile d'accéder à n'importe quel pixel d'une image. Voici un exemple de code qui démontre comment accéder à tous les pixels pour en faire une moyenne :

Exemple de code

```
int index  = 0; //Première camera trouvée.
var image  = bsdk.cameras[index].Image.GetImage();
int width  = bsdk.cameras[index].Image.width;
int height = bsdk.cameras[index].Image.height;

double pixelSum = 0.0;

for (int i = 0; i < height; i++)
{
  for (int j = 0; j < width; j++)
  {
     pixelSum += image[i * height + j];
  }
}

//Faire une moyenne.
pixelSum /= (width * height);
```

### 3.3. BCamSettings


***Déclaration de classe***

```
class BCamSettings
```

***Attributs***

```
float exposureTime;
```

Cette valeur peut être paramétrée entre 0,6 ms et 200 ms.

**Évènements**

La classe BCamSettings ne contient pas d'évènements (Events).

### 3.4. BCamProperties


BCamProperties correspond aux propriétés internes de la caméra *Beamage*. Contrairement à la classe BCamSettings, rien ne peut être modifié dans cette classe puisque tout est en mode lecture seule.

***Déclaration de classe***

***Attributs***

La classe BCamProperties ne contient pas d'attributs.

***Événements***

La classe BCamProperties ne contient pas d'évènements (Events).

***Fonctions***

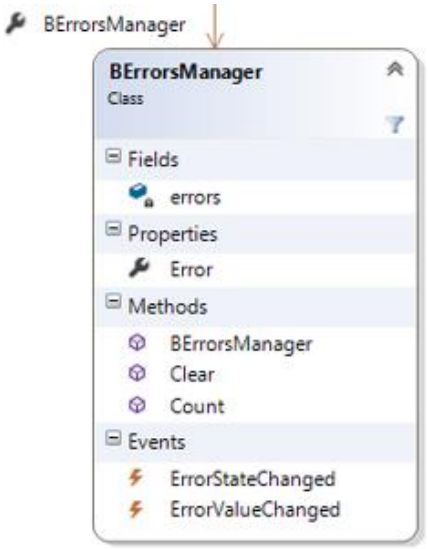
```
bool Is4mSensor()
```

Cette fonction indique si la caméra *Beamage* est de type 4M ou 3.0. Si la caméra est un modèle Beamage-4M, la fonction retournera la valeur *True*. Si c'est un modèle Beamage-3.0, elle retournera la valeur *False.*

```
string GetSerialNumber()
```

Cette fonction retourne une chaîne de caractères correspondant au numéro de série de la caméra *Beamage*.

### 3.5. BErrorsManager




La classe BErrorsManager contient des événements auxquels on peut s'abonner. Cette classe viendra en aide aux programmeurs qui apprennent à se servir du kit de développement *Beamage SDK* en donnant davantage de renseignements sur les causes d'erreur possibles. Pour recevoir les messages d'erreurs, il faut obligatoirement s'abonner à l'EventHandler.

***Déclaration de classe***

```
class BErrorsManager
```

***Attributs***

```
string Error;
```

Le *string* contenant le message d'erreur avec tous les détails sur les erreurs rencontrées durant l'exécution du code.

***Événements***

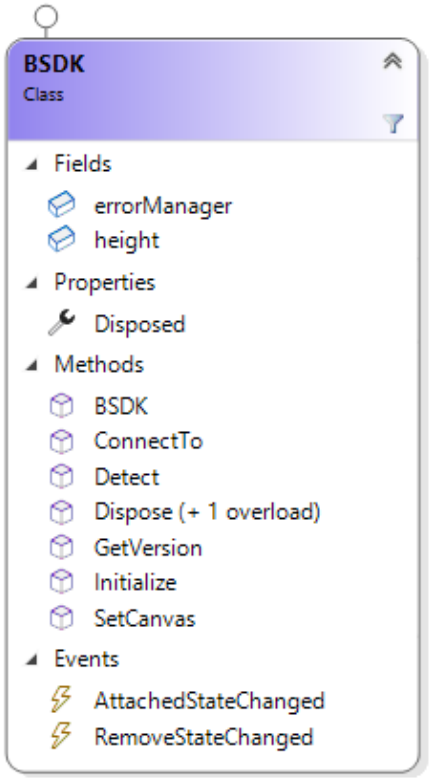
```
event EventHandler ErrorStateChanged;
event EventHandler ErrorValueChanged;
```

Voici comment s'abonner aux événements de cette classe :

Exemple de code

```
        // Assigne un gestionnaire d'erreurs pour les erreurs provenant du BSDK.
        // Le gestionnaire d'erreurs va retourner une erreur si la caméra détecte quelque chose d'inhabituel.
        bsdk.errorManager.ErrorStateChanged += new EventHandler(errorsEvent);

and
    private void errorsEvent(object sender, EventArgs e)
    {
        // Messages d'erreurs du gestionnaire d'erreurs.
        string error = ((BErrorsManager)sender).Error;
        MessageBox.Show(error);
    }
```

### 3.6. BSDK




BSDK est la classe principale du kit de développement *Beamage SDK*. Voir la section Débuter avec le code pour obtenir plus de détails sur son fonctionnement.

***Déclaration de classe***

`BSDK()`
Principale classe du kit de développement *Beamage SDK*.

***Événements***

`event EventHandler AttachedStateChanged;`
Ce gestionnaire d'évènements (*EventHandler*) se déclenchera chaque fois qu'une caméra Beamage sera branchée physiquement à l'ordinateur. Cela permet à un utilisateur de connecter une caméra en cours d'utilisation du logiciel ou encore d'opérer certains changements si une nouvelle caméra est branchée.

`event EventHandler RemoveStateChanged;`
Ce gestionnaire d'évènements (*Event Handler*) se déclenchera chaque fois qu'une caméra sera déconnectée physiquement de l'ordinateur. Cela permettra d'opérer certains changements si une caméra se déconnecte ou de vérifier si votre caméra est toujours accessible. Cette fonction pourrait aussi permettre de diagnostiquer un port USB déficient.

Exemple de code

```
// Assigne un événement pour la connexion ou la déconnexion d'une caméra.
bsdk.AttachedStateChanged += new EventHandler(attachedEvent);
bsdk.RemoveStateChanged += new EventHandler(removeEvent);
```

***Fonctions***

`void ConnectTo(int _Index)`
Permet de se connecter à une caméra Beamage à l'aide de son index. L'index de la caméra est sa position dans la liste. Ainsi, pour connecter la première caméra, utilisez l'index 0.

`public void Detect()`
Cette fonction détecte et initialise toutes les caméras Beamage connectées physiquement à l'ordinateur et les ajoute dans la liste des caméras. Si une caméra était déjà connectée avant l'appel de cette méthode, elle le demeurera par après.

`public string GetVersion()`
Cette fonction retourne en chaine de caractères le numéro de version du SDK Beamage.

`public void Initialize()`
Cette fonction fait appel à la fonction `Detect()` pour initialiser les caméras Beamage connectées.

`public void SetCanvas (int _height)`
Définissez la valeur pour changer la taille de la hauteur du canevas. La valeur de la largeur n'est pas modifiable et sa valeur par défaut est 2048.

`void Dispose()`
La classe BSDK implémente IDisposable et doit avoir une fonction Dispose.

gentec-eo
PARTENARIAT de PRÉCISION

WWW.GENTEC-EO.FR

# CHEF DE FILE EN MESURE LASER **DEPUIS 1972**


PUISSANCE ET ÉNERGIE LASER


PROFILOMÉTRIE LASER

MESUREURS THZ


CANADA

445 St-Jean-Baptiste, Suite 160
Quebec, QC, G2E 5N7
CANADA

T (418) 651-8003
F (418) 651-1174

info@gentec-eo.com

ÉTATS-UNIS

5825 Jean Road Center
Lake Oswego, OR, 97035
USA

T (503) 697-1870
F (503) 697-0633

info@gentec-eo.com

JAPON

Office No. 101, EXL111 building,
Takinogawa, Kita-ku, Tokyo
114-0023, JAPAN

T +81-3-5972-1290
F +81-3-5972-1291

info@gentec-eo.com

CENTRES DE CALIBRATION

- 445 St-Jean-Baptiste, Suite 160
  Quebec, QC, G2E 5N7, CANADA
- Werner von Siemens Str. 15
  82140 Olching, GERMANY
- Office No. 101, EXL111 building,
  Takinogawa, Kita-ku, Tokyo
  114-0023, JAPAN