Training Application for Industries by Means of Virtual Reality Technology

MIHALACHE GHINEA
Department of Machines and Manufacturing Systems
University POLITEHNICA of Bucharest
Splaiul Independenţei, 313, sector 6, 060042, Bucharest
ROMANIA
ghinea2003@yahoo.com, www.pub.ro

GICU CĂLIN DEAC
Department of Machines and Manufacturing Systems
University POLITEHNICA of Bucharest
Splaiul Independenţei, 313, sector 6, 060042, Bucharest
ROMANIA
gicu.deac@gmail.com, www.pub.ro

CRINA NARCISA GEORGESCU
Department of Machines and Manufacturing Systems
University POLITEHNICA of Bucharest
Splaiul Independenţei, 313, sector 6, 060042, Bucharest
ROMANIA
crina.deac@gmail.com, www.pub.ro

Abstract: One of the pylons of the Industry 4.0 is augmented and virtual reality. It can improve the perception on the industrial processes in real time and more. The purpose of these research is to obtain a starting point regarding the actual abilities to enlarge the use of VR HMDs (e.g.: HTC Vive, Oculus Rift, etc.), assembling into a unique application, Virtual and Augmented Reality with Gesture Interaction. The project relies on the Leap Motion controller. This controller have an integrated infrared depth camera wich can be used as an input video feed for stereoscopic view on Oculus Rift, in the same time, projecting the raw images of your hands into a 3D mesh that can interact with other objects in real-world space (in our case a 3D model of a complex product). The pass through from the twin Leap Motion cameras covers the part of the mesh such that each camera assures a distinct view of the hands, letting you see the actual depth. We can interact with the 3D model and show some functional animations. We present a part of a industrial application of VR used into a common industry (door locks manufacturing industry), which can be useful for research and for training and advertising too.

Keywords: virtual reality, augmented reality, Oculus Rift, Leap Motion, Industry 4.0

1 Introduction

Augmented reality (AR) is defined as a live direct or indirect view of the real-world environment whose elements are improved (supplemented or augmented) by computer-generated sensory input such as GPS data or sound, video, graphics, [8][11][14].

AR is linked to a broader concept named mediated reality, where the view of reality is modified (it can be even diminished rather than augmented) by a computer. This technology enhances the current user perception of the reality. [1][2]. In exchange, virtual reality simulates the real world. Augmentation is producing in real-time and in semantic context with environmental elements. [3][4].

The advanced AR technology including object recognition and computer vision makes possible that enclosing elements of the real words user become intelligent, interactive.

Information about the elements of the environment are superimposed on the real world

[1][3][4][5][6][7]. This information can be virtual or real, e.g. seeing other real sensed or measured information such as electromagnetic radio waves overlaid in exact alignment with where they actually are in space [9][10].

The new devices such as e.tablets, smartphones or Augmented Reality eye-wear has encouraged the development of applications especially in the area of industrial maintenance and informative or educational geolocation. Uses of virtual reality in systems such as parts analysis and simulation, staff support, layout and construction planning or supervising can no longer be set in the future. Augmented Reality as a Virtual Support to Industries.

The development of Augmented Reality Technology to assist industries with precise site or field information in real-time is today a foreseeable reality to be used in almost any domain. For instance, a project management in building and construction will be more easily and safely carried out when site managers can virtually view and monitor work in progress in real time through Augmented Reality markers placed on parts or equipment being built. Pointing a camera to factory on-site piece of equipment can match it to the digital map of the plant and verify it is in its designated location, not only freeing the staff from cumbersome paper layout plans but also providing the operators with virtual reality contextual information.

Industry decision makers can make timely decisions when management foresees how a piece of equipment or a machine once built will fit in its final environment, by merely looking at the superimposition of field data fed through Augmented Reality Systems.

1.1 Enhancing productivity and work behavior

It is now possible to mix Augmented Reality with geolocation, recognition and tracking technologies. The instant information coupled with enhanced perception will ensure that Augmented Reality systems play a big role in how people and companies work in the future.

Milgram defined a continuum of Real to Virtual environments, where Augmented Reality is one part of the general area of *Mixed Reality* (Figure 1). In both Augmented Virtuality and Virtual Environments (a.k.a Virtual Reality), the surrounding environment is virtual, while in AR the surrounding environment is real. This survey focuses on Augmented Reality and does not cover Augmented Virtuality or Virtual Environments [12][13][14].

Until recently, most AR interfaces were based on the desktop metaphor or used designs from Virtual Environments research. One main trend in interaction research specifically for AR systems is the use of heterogeneous designs and tangible interfaces. Heterogeneous approaches blur the boundaries between real and virtual, taking parts from both worlds. Tangible interfaces emphasize the use of real, physical objects and tools. Since in AR systems the user sees the real world and often desires to interact with real objects, it is appropriate for the AR interface to have a real component instead of remaining entirely virtual.

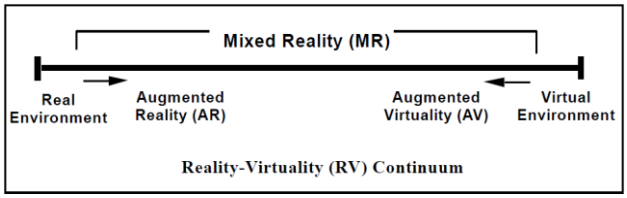

Fig.1 Milgram's Continuum of real to virtual environments

Virtual Reality (VR is all about the creation of a virtual world that users can interact with. Users are isolated from the real world while immersed in a full synthetic environment, as far as the immersive experience is achieved by the wearing of a VR helmet (e.g.: Oculus Rift).

On the other hand, Augmented Reality (AR) blends virtual reality contents with the real world. As a result, users can interact with virtual contents while continuing to be in touch with the real life around them. This experience is achieved by the wearing of AR glasses (e.g.: Google Glasses).

Technically speaking, the main difference between the two systems seems to be a transparency mask issue. VR scenes are completely opaque over the entire display (i.e. you cannot see the real world around you). AR glasses are transparent (you see the real environment) and pixels where virtual contents are drawn have an opacity value in order to overwrite the real world light information. From this point of view, VR is simply a transparency mask limit case of AR, where transparency has a null value over the entire display.

Often it seems that VR and AR are two different worlds that do not overlap. But what if we imagine VR and AR on the same next generation device? Why do not have a unique system that could provide both AR and VR experiences? Well, it is not so simple as it looks like.

2 Research activity

The purpose of these research is to provide a practical guide of the real possibilities to extend the way of using **VR HMDs** (e.g.: Oculus Rift, HTC Vive etc.), gathering together Virtual Reality, Augmented Reality and Gesture Interaction in a unique project (Figure 2).

The project relies on the **Leap Motion** controller, a small puck-like device that tracks the motion of your arms and hands. This controller can be mounted on the HMD.

Fig.2 Leap motion controller mounted on Oculus Rift

Leap motion controller have an integrated infrared depth camera. This video camera can be used as an input video feed for stereoscopic view on an HMD like **Oculus Rift**, in the same time, projecting the raw images of user hands into a 3D mesh that can interact with other objects in real-world space. The visible part of the mesh is covered by the passthrough from the twin Leap Motion cameras. Each camera provides a separate view of the user hands, so the user can see the image with actual depth.

Using this approach has a powerful impact in VR because your real hands can now actually pass through (or disappear behind) virtual objects. The hands can interact properly with other 3D objects in the scene because they are 3D and have visual capabilities that you expect. Using AR hands also reduces jitter effects, since there's no longer an artificially generated rigged hand wich is unresponsive when the hands are not recognized propperly.. While the hand image in VR might shift or become transparent, it won't suddenly shift in the wrong direction as a rigged hand.

The **Image Hands** are also designed to provide users with dynamic feedback on the Leap Motion Controller's tracking confidence [1]. When your hand assumes a high-confidence pose, it will glow blue. The glow disappears as confidence values drop (Figure 3).

The Leap Motion controller uses infrared stereo cameras as tracking sensors. The images provided by this twin cameras can be accessed using the **Controller.Images** or **Frame.Images** functions [1][16]. These functions provide an **ImageList** object, containing the **Image** objects. **Controller.Images** provides the most recent set of images and **Frame.Images** provides the set of images analysed to create that frame and can be slightly obsolete compared to the images returned directly by the Controller.

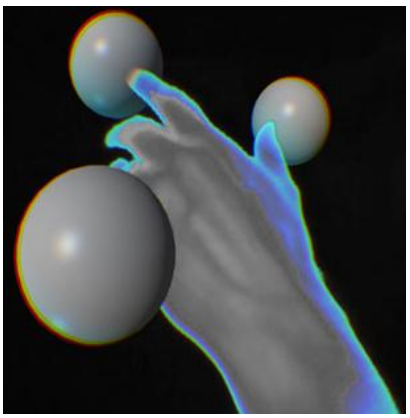

Fig. 3 Image hands preview

When it obtains an image from one of the cameras, a grid highlighting the significant and complex distortion is superimposed on the image (Figure 4). The images can be used for: **Head-Mounted Display** video pass-through, **Augmented Reality**, **Computer Vision**. The Image API (Application Programming Interface) provides a buffer containing the sensor brightness values and a buffer containing the camera calibration map, which can be used to correct lens distortion and other optical imperfections in the image data[1][16].

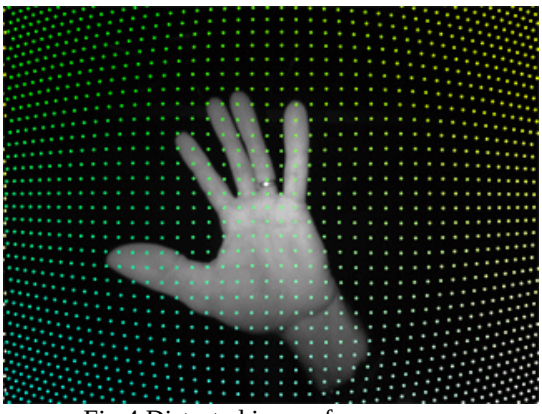

Fig.4 Distorted image from camera

2.1 Image Distortion

When a ray of light enters one of the Leap Motion cameras, the lens bends the light ray so that it hits the sensor. The sensor records it as a greyscale value of brightness at a relative pixel location. Because no lens is perfect, the ray of light does not land on the sensor in the perfect optically spot. The calibration map provides usefull data to correct this imperfection by calculating the true angle of the original ray of light. It is posible to triangulate the

3D location of a feature identified in both images. Tthe calibration map corrects lens distortion but not correct the perspective distortion [1][15].

2.2 Image Ray Correction

We can get the camera Raw images using:

```
controller.setPolicy(Leap::Controller::POLICY_IMAGES);
```

After image orientation, getting raw images and getting calibration map, it must make the correction of the image ray.

The raw image distortion can be corrected in two ways [1][15][16]:

- Using the **Image.Warp()** and **Image.Rectify()** functions.
- Using the data from the **Image.Distortion** buffer directly.

The **Warp()** and **Rectify()** functions are the simpler method, but is relatively slow because it process each pixel individually on the CPU. The Distortion buffer is designed to be used with a GPU shader program and can correct the entire raw image while maintaining a good application frame rate.

We use a 3D object modelled in Catia and imported in Cinema 4D where we created all the animations. We have two scripts, one script for reading point clouds of the 3D model and one script for extract this points, for interaction with the model with image hands [1].

3 Finding improvements

With a satisfactory precision, CAD models can be imported into certain programs for 3D animation. This fact is more and more important for applications in industry, medicine, biology (to name only few of them), as they need quite realistic image.

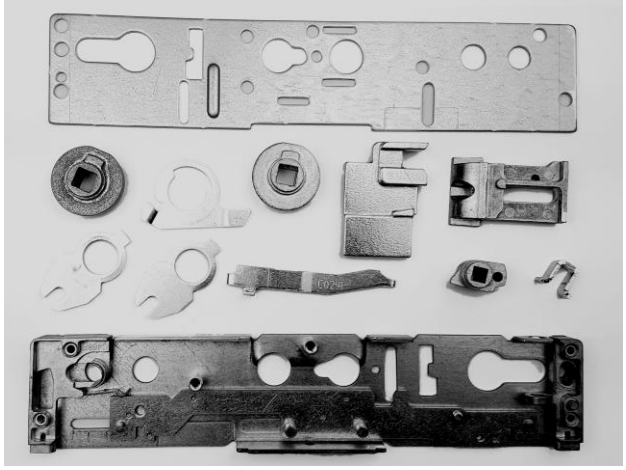

Fig.5 The parts of the door lock system

Thus, Image API applications become increasingly useful to outcomes not necessarily spectacular but closer to reality. In figure 5, the real image of a door lock system, with all its parts, is presented.

Conceiving a CAD model (Figure 6) by means of an application such as CATIA could lead to a virtual representation of the real model with a 100% accuracy. Although 3D scanning could be a solution, as well, even faster and cheaper, the incosistencies that introduce the real mode must be taken into consideration.

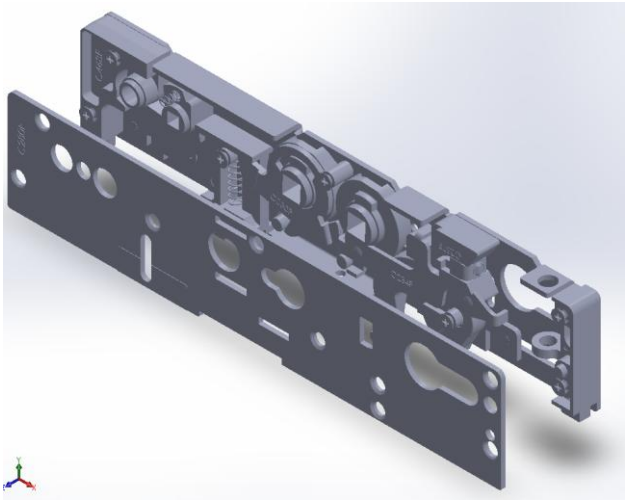

Fig.6 The CAD model of the door lock system

The project was developped in Unity 3D, using the Leap Motion Core Unity Assets and the LMHeadMountedRig prefab wich include a camera and hand controller setup wich replace the standard camera from scene (Figure 7). The scripts from LMHeadMountedRig adjust automaticaly the position of stereo camera in order to correct interpupillary distance and compensate the video lag in augmented scenes. For LMHeadMountedRig we use the AR World AR hands option. The ThresholdOverlay shader can be asigned to an overlay quad composited over the scene to display the hands even if are not recognized as hands.

In addition to prefabs and scripts included in the asset package it is possible to access tracking data directly by Leap Motion API:

```
using UnityEngine;
using System.Collections.Generic;
using Leap;
public class LeapBehavior : MonoBehaviour {
  LeapProvider provider;
```

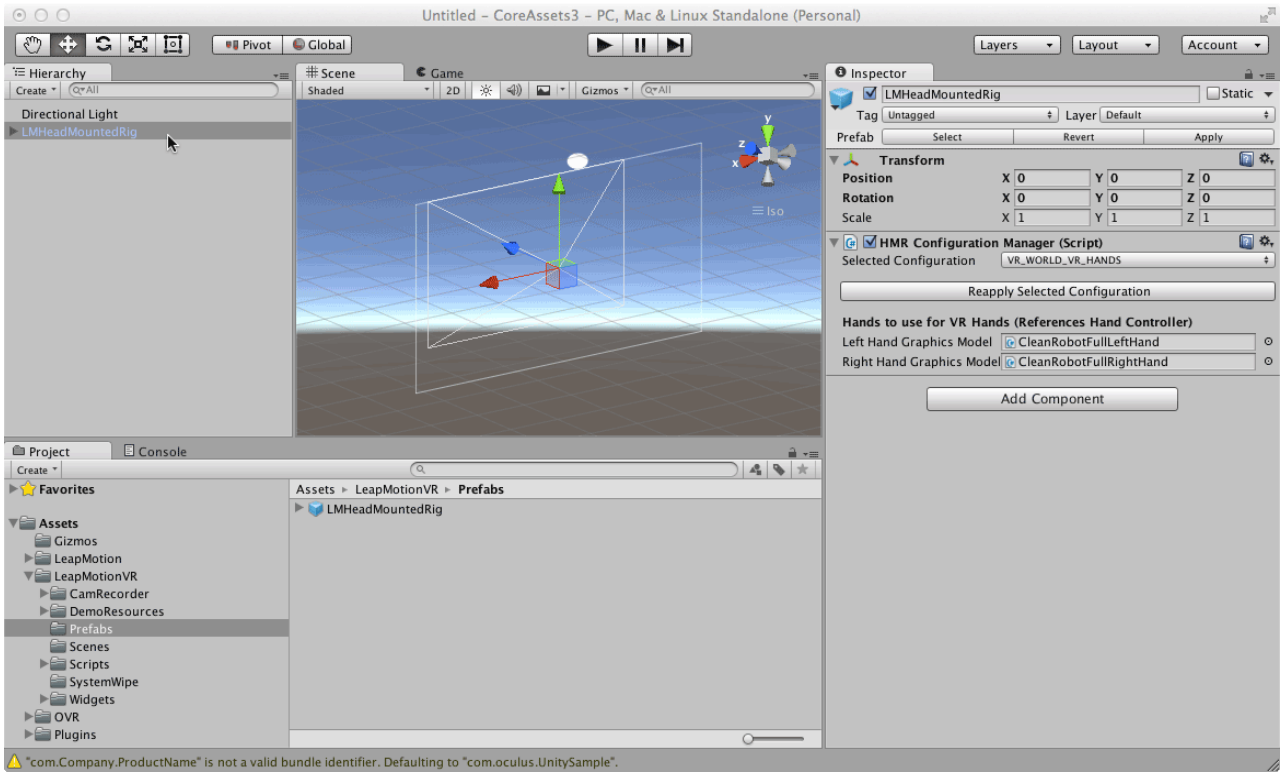

Fig.7 Unity Interface LMHeadMountedRig prefab

```
  void Start ()
  {
      provider =
FindObjectOfType<LeapProvider>() as
LeapProvider;
  }
  void Update ()
  {
 Frame frame = provider.CurrentFrame;
 foreach (Hand hand in frame.Hands)
     {
       if (hand.IsLeft)
       {
          transform.position =
hand.PalmPosition.ToVector3() +
hand.PalmNormal.ToVector3() *
(transform.localScale.y * .5f + .02f);
transform.rotation = hand.Basis.Rotation();
       }
     }
  }
}
```

To a better immersion of the user in the virtual space we wanted to use the image provided by the Leap Motion Camera as a passthrough texture.

We have to functions to display and correct distorsions:

ImagePassthrough::updateImage()and ImagePassthrough::updateDistortion()

```
void ImagePassthrough::updateDistortion(int idx,
const Leap::Image& image) {
 std::shared_ptr<Leap::GL::Texture2>& distortion
= m_Distortion[idx];
 const float* data = image.distortion();
 const int width = image.distortionWidth()/2;
 const int height = image.distortionHeight();
 const int bytesPerPixel = 2 * sizeof(float); // XY
per pixel
 const size_t numBytes = static_cast<size_t>(width
* height * bytesPerPixel);
 Leap::GL::Texture2PixelData pixelData(GL_RG,
GL_FLOAT, data, numBytes);
 if (!distortion || numBytes !=
m_DistortionBytes[idx]) {
   Leap::GL::Texture2Params
params(static_cast<GLsizei>(width),
static_cast<GLsizei>(height));
```

```
    params.SetTarget(GL_TEXTURE_2D);
    params.SetInternalFormat(GL_RG32F);

params.SetTexParameteri(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);

params.SetTexParameteri(GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

params.SetTexParameteri(GL_TEXTURE_MAG_FILTER, GL_LINEAR);

params.SetTexParameteri(GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    distortion = std::shared_ptr<Leap::GL::Texture2>(new Leap::GL::Texture2(params, pixelData));
    m_DistortionBytes[idx] = numBytes;
  } else {
    distortion->TexSubImage(pixelData);
  }
}

void ImagePassthrough::updateImage(int idx, const Leap::Image& image) {
  GLenum format = (image.width() == 640) ? GL_LUMINANCE : GL_RGBA;
  m_Color = (format == GL_RGBA);

  std::shared_ptr<Leap::GL::Texture2>& tex = m_Textures[idx];
  const unsigned char* data = image.data();
  const int width = image.width();
  const int height = image.height();
  const int bytesPerPixel = image.bytesPerPixel();
  const size_t numBytes = static_cast<size_t>(width * height * bytesPerPixel);
  Leap::GL::Texture2PixelData pixelData(format, GL_UNSIGNED_BYTE, data, numBytes);
  if (!tex || numBytes != m_ImageBytes[idx]) {
    Leap::GL::Texture2Params params(static_cast<GLsizei>(width), static_cast<GLsizei>(height));
    params.SetTarget(GL_TEXTURE_2D);
    params.SetInternalFormat(format);

params.SetTexParameteri(GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);

params.SetTexParameteri(GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

params.SetTexParameter(GL_TEXTURE_MAG_FILTER, GL_LINEAR);

params.SetTexParameteri(GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    tex = std::shared_ptr<Leap::GL::Texture2>(new Leap::GL::Texture2(params, pixelData));
    m_ImageBytes[idx] = numBytes;
  } else {
    tex->TexSubImage(pixelData);
  }
}
```

We use the **LeapProvider** object to transform the frame data into the proper frame of reference to match the hands displayed in a scene.

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

/// <summary>
/// Enables rescaling of an object while preventing rescaling of specified child objects
/// </summary>
public class CompensatedRescale : MonoBehaviour
{
  [Header("Scale-Invariant Children")]
  public List<Transform> compensated;
  [Header("Control Keys")]
  public KeyCode unlockHold = KeyCode.RightShift;
  public KeyCode resetScale = KeyCode.R;
  public KeyCode increaseScale = KeyCode.Equals;
  public KeyCode decreaseScale = KeyCode.Minus;
  [Range(0,1)]
  public float decreaseFactor = 0.625f; //40 mm CFS / 64 mm IPD

  [Range(0.25f,4f)]
  public float newScaleFactor = 1f;
  private float oldScaleFactor = 1f;

  private Vector3 initialScale;

        // Use this for initialization
        void OnEnable () {
  initialScale = transform.localScale;
```

```
        }

 void OnDisable () {
  ResetScale ();
 }

        // Update is called once per frame
        void Update () {
          if (unlockHold != KeyCode.None &&
     !Input.GetKey (unlockHold)) {
   return;
  }
  if (Input.GetKeyDown (resetScale)) {
   ResetScale();
   return;
  }
  if (Input.GetKeyDown (increaseScale)) {
   IncreaseScale();
   Debug.Log ("IncreaseScale");
   return;
  }
  if (Input.GetKeyDown (decreaseScale)) {
   DecreaseScale();
   Debug.Log ("DecreaseScale");
   return;
  }

  if (oldScaleFactor != newScaleFactor) {
   ApplyRescale (newScaleFactor /
oldScaleFactor);
   oldScaleFactor = newScaleFactor;
   Debug.Log("newScaleFactor = " +
newScaleFactor);
  }
 }

 public void ResetScale() {
  oldScaleFactor = newScaleFactor = 1f;

  float multiplier = (
   (initialScale.x / transform.localScale.x) +
   (initialScale.y / transform.localScale.y) +
   (initialScale.z / transform.localScale.z)
   ) / 3f;
  ApplyRescale(multiplier);
 }

 public void IncreaseScale() {
  ApplyRescale(1f / decreaseFactor);
 }

 public void DecreaseScale() {
  ApplyRescale(decreaseFactor);
 }

 void ApplyRescale(float multiplier) {
  transform.localScale *= multiplier;
  foreach (Transform child in compensated) {
   child.localScale /= multiplier;
  }
 }
}
```

To read the imported 3D model we have the read function:

```
using UnityEngine;
using System.Collections;

public class test_different_file_read :
MonoBehaviour {
  float[,] imported_model;
  // Use this for initialization
  void Start () {

       }
  public void DoSomething()
  {
    imported_model =
ReadPointCloudFile.stored_point_cloud;

    for (int i = 0; i < 100; i++)
    {
      Debug.Log(imported_model[i, 2]);
    }
  }
        // Update is called once per frame
        void Update () {
        }
}
```

To start the animation of the 3D model we use a script wich trigger the animation by pressing the space key:

```
public bool animation_bool;
        void Update()
                {
           if(animation_bool == true)
                {
             animation.Play("asa-abloy");
                }
          if(Input.GetButtonDown("space"))
                {
            animation_bool = true;
                }
             }
```

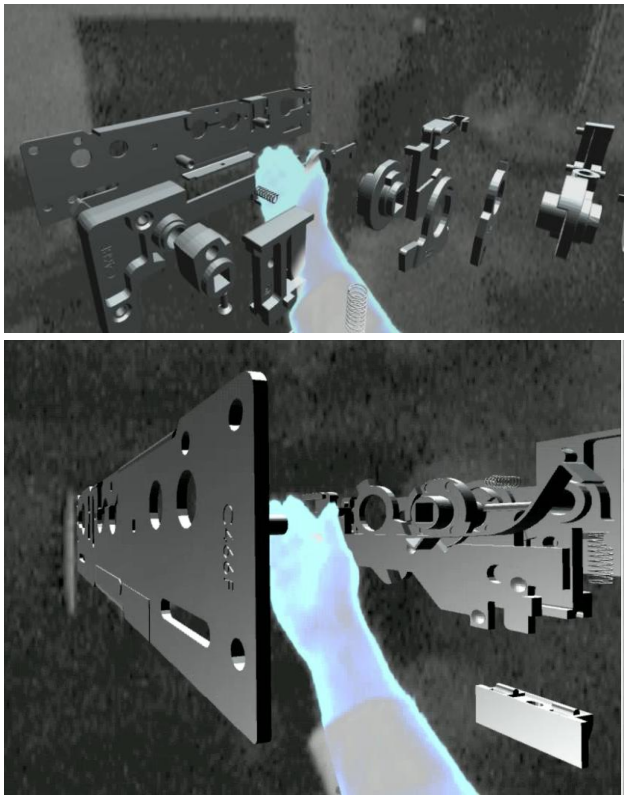

Fig. 8 3D model moved in VR by image hands

The present article exibiths the collective endeavors of the reserch team aiming to achieve a post-processing software that enables the virtual immersion as being the real one. Unfortunately, only by using virtual immersion equipment one can understand how impressive the advancement is. The images in Figures 8, are the ones precessed by means of the above mentioned method.

4 Conclusions

All over the world, VR technology is used today for training applications in a variety of process industries, and enables personnel subjection to simulated hazardous situation in a safe, highly visual and interactive way.

Customized simulations of plants layouts, dynamic processes and comprehensive virtual environments can be set up and allow users to move within the virtual plants or systems, making operational decisions and investigating processes at a glance.

Our aim is to couple activity of lab AVRENG (Augmented & Virtual Reality for Engineering), from University POLITEHNICA of Bucharest with virtual reality and virtual environments applications for future industrial workspaces. We want to gather expertise from partner members and determine the future research agenda for the development and use of virtual reality (VR) technologies. The working team on Education and Training is specifically focused on understanding how VR is used to support learning in educational and industrial contexts.

This paper represents the very first steps of VR technology currently in use or development for training in industry. It remains important to identify potential future development of VR training applications and also to overcome the existent barriers.

For the next version of our application, we think about implementing a color stereoscopic camera to provide our industrial partners with an improved experience and a higher accuracy.

References:

[1] Mihalache Ghinea, Gicu Calin Deac, Crina Narcisa Georgescu. (2016) *Improving the Quality of Image in Virtual Reality Applications for Industry*. WSEAS Conference, 3rd International Conference onAeronautical and Mechanical Engineering (AEME '16) Bern, Switzerland December 17-19, 2016, published in:International Journal of Computers, **1**, 284-289, ISSN: 2367-8895

[2] Graham, M., Zook, M., and Boulton, A. , *Augmented reality in urban places: contested content and the duplicity of code*, Transactions of the Institute of British Geographers, DOI: 10.1111/j.1475-5661.2012.00539.x 2012.

[3] Steuer, J. *Defining Virtual Reality: Dimensions Determining Telepresence*, Department of Communication, Stanford University, 15 Oct., 1993.

[4] *** Introducing Virtual Environments National Center for Supercomputing Applications, University of Illinois.

[5] Chen, B.X. *If You're Not Seeing Data, You're Not Seeing*, Wired, 25 August 2009.
[6] Maxwell, K. *Augmented Reality*, Macmillan Dictionary Buzzword.
[7] Azuma, R. *A Survey of Augmented Reality Presence: Teleoperators and Virtual Environments*, pp. 355–385, August 1997.
[8] Zhanpeng, H.,Pan H., et al. *Mobile augmented reality survey: a bottom-up approach*.
[9] *Phenomenal Augmented Reality*, IEEE Consumer Electronics, Vol. 4, No. 4, October 2015.
[10] Archibald, C., Petriu, E. *Time-frequency perspectives, with applications*, in Advances in Machine Vision, Strategies and Applications, World Scientific Series in Computer Science: Volume 32.
[11] Metz, R. *Augmented Reality Is Finally Getting Real* Technology Review, 2 August 2012.
[12] Fleet Week: *Office of Naval Research Technology- Virtual Reality Welder Training*, eweek, 28 May 2012.
[13] Rolland, J. Baillott, Y. Goon, A..A., *Survey of Tracking Technology for Virtual Environments*, Center for Research and Education in Optics and Lasers, University of Central Florida.
[14] Azuma, R., Balliot, Y., Behringer, R., Feiner, S., Julier, S., MacIntyre, B. *Recent Advances in Augmented Reality*, Computers & Graphics, November 2001.
[15] https://developer.leapmotion.com/documentation/csharp/index.htm
[16] http://iiif.io/api/image/2.0