Markerless Tracking Solutions for Augmented Reality on the Web

João Paulo S. do M. Lima, Pablo C. Pinheiro, Veronica Teichrieb, Judith Kelner
Virtual Reality and Multimedia Research Group (GRVM) – Informatics Center (CIn)
Federal University of Pernambuco (UFPE)
{jpsml, pcp, vt, jk}@cin.ufpe.br

Abstract

This paper presents tracking software infrastructure developed for allowing the creation of web targeted augmented reality applications based on natural features of the real scene. A texture based markerless tracking by detection technique for planar objects was used with descriptor based (SIFT and SURF) and classifier based (Ferns) feature matching methods. One server-side tracking infrastructure was implemented using the Mammoth open source Flash media server. In addition, two client-side tracking solutions were developed, each one based on a different browser plug-in (OSAKit and Silverlight). The implemented solutions were evaluated using real data regarding metrics such as performance and load time. Client-side tracking provided better results, with the OSAKit solution having higher frame rates, while the Silverlight solution showed to be more flexible and platform independent.

1. Introduction

The desktop platform is the target environment most commonly addressed when developing Augmented Reality (AR) systems. However, depending on the requirements of an AR application, the use of different execution platforms may be necessary. If the system has to be published to several users, the web platform shows to be more adequate, where the application is executed through the Internet in a web browser. In the last years, AR technology has been often used in web applications, most of the times for marketing purposes. For example, important results have been obtained by the advertising piece developed by the German company Metaio for the launch of the new model from the British car manufacturer MINI [1]. An application was created where the user is capable of visualizing a virtual replica of the car over a print ad. The advertisement was cited in more than 100 magazine/blog articles and the making-of video was seen by more than 200,000 people on the Internet, which highlights the success of the initiative.

The use of markerless tracking, which is based on natural features of the scene, has also been gaining more space on web targeted AR applications for advertising. The media used in this kind of application needs to be as appealing as possible in order to catch consumers' attention. Markerless tracking satisfies this requirement, since the idea of having a real scene augmented with virtual objects without any artificial elements such as markers added to the environment is very attractive. In addition, the own package of the product being advertised can be tracked and augmented with virtual elements.

In this context, this paper aims to present the implementation and evaluation of some solutions regarding markerless tracking for web targeted AR. The criteria used in the evaluation of the solution include performance, load time, platform independency and technology diffusion. The keypoint based technique described in [2] was adapted to work on the Internet. Different approaches for feature matching were utilized (descriptor based [3][4][5] and classifier based [6]). Three different technologies have been evaluated: Mammoth [7], OSAKit [8] and Silverlight [9]. When using Mammoth, tracking takes place at server-side, while the utilization of OSAKit and Silverlight allows tracking to occur at client-side.

This paper is organized as follows. Section 2 presents related work regarding web AR. Section 3 describes the keypoint based markerless tracking technique used in this work. Section 4 details the three markerless tracking solutions for the web developed in this work. A discussion about the results obtained is presented in Section 5. Section 6 draws some conclusions and future work.

2. Related work

The first more concrete initiatives towards marker based AR for the Internet occurred when the

FLARToolKit library was created [10]. FLARToolKit is a port of the well-known ARToolKit marker tracking library to ActionScript, which is the language utilized in the development of Flash applications for the web. By using FLARToolKit together with Papervision3D, which consists in a graphics library for Flash [11], it is possible to develop applications that run on the client's browser, requiring only the Flash plug-in. A number of marketing actions on the web have been using FLARToolKit and Papervision3D, such as a campaign by General Electric, illustrated in Figure 1, where it is possible to visualize a wind turbine over a marker and interact with it by blowing into a microphone [12].

Figure 1. Marker based AR for the Internet using FLARToolKit

The Metaio company developed Unifeye Viewer, a proprietary plug-in for Shockwave that allows the utilization of markerless AR applications on the Internet [13]. An application similar to the General Electric's one shown previously was developed using this plug-in [14], but tracking a magazine cover instead of a marker, as shown in Figure 2.

Figure 2. Markerless AR for the Internet using Unifeye Viewer

A similar approach was adopted by the Total Immersion company when creating D'Fusion@Home, which also consists of a proprietary plug-in for web targeted AR applications [15]. It was utilized in some advertising pieces, such as the Transformers II movie promotion where the user's head is augmented with a virtual helmet [16].

Finally, the Imagination company developed the flash augmented reality engine (flare), which is a markerless tracking solution for the Internet based on Flash, as well as FLARToolKit, demanding that the client has the Flash plug-in installed [17]. An example of application that uses this library is the The Red Bulletin website, where it is possible to watch animations and videos superimposed on pages of a magazine [18].

Figure 3. Markerless AR for the Internet using the flare library

3. Markerless tracking technique

The basic difference between markerless and marker based tracking is that the first does not require artificial markers put on the environment in order to estimate camera pose. Markerless tracking can, in theory, use any part of the scene as a reference to position virtual objects (in practice, some restrictions may apply, depending on the specific technique used) [19].

Several algorithms for markerless tracking are available in the literature **Erro! Fonte de referência não encontrada.**. Some of them require a previous knowledge about the environment, such as a digital model of a real object to be tracked, the texture of a region of the scene, etc. These methods are known as model based. Other techniques can make an online estimate of the topology of the environment, using only different frames captured from the scene by a moving camera for registration of the virtual objects. These techniques are called Structure from Motion

(SfM) and are often more complex and require more processing power to achieve real-time frame rates.

Since this work aims to develop some web targeted markerless AR solutions, the keypoint based technique was chosen for tracking purposes, which is a model based technique described in [2]. It is a tracking by detection technique, being able to detect the target object at every frame without any previous estimate of its pose, allowing automatic initialization and recovery from failures. Commonly, web applications are utilized by novice users that do not have sufficient technical knowledge to perform manual initialization, making it a suitable technique to such applications.

This technique is based on texture information extracted from the scene. An offline training phase is needed in order to acquire knowledge about the object to be tracked. A set of 2D object features that are invariant to scale, illumination and viewpoint are obtained, together with their corresponding 3D position in the object model. At runtime, two steps are performed: feature matching and pose calculation. In the feature matching phase, invariant features are extracted from the current frame and matched with the acquired knowledge base, resulting in 2D-3D correspondences. In the pose calculation phase, these correspondences are used to compute the current pose without any previous estimate. The feature matching and pose calculation steps are described in the following subsections.

3.1. Feature matching

The feature matching techniques used in the case studies performed in this work (and presented in section 4) search for correspondent points in the current frame and in the knowledge base. Only points that are highly descriptive invariant features, called keypoints, are tested. Figure 4 illustrates the feature matching step of the keypoint based method. The left image is a reference image used in the training phase and the right image is the current frame of the tracking sequence. The dots denote features extracted from both images and the lines link the correspondences.

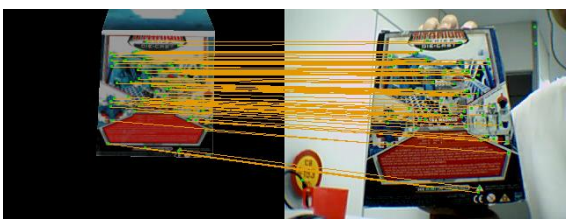

Figure 4. Feature matching example

These techniques can be further classified in descriptor based and classifier based, both described below.

3.1.1. Descriptor based

The keypoint matching approach used in algorithms of this class is based on the similarity of descriptors created from patches around the keypoints.

In this work, two descriptor based techniques were applied in the context of web targeted markerless AR: SIFT [3][4] and SURF [5]. These techniques use high-dimensional vectors as descriptors. An evaluation of the use of such descriptors for model based tracking can be found in [2].

In order to make the search for correspondents more efficient, a kd-tree [3] is used to store the descriptors of the keypoints from the knowledge base. Then, for each extracted keypoint of the current frame, it is executed a Best Bin First search [3] in the kd-tree to find its match (if it has one). This search returns the two nearest neighbors of the keypoint. By verifying the Euclidian distance between the neighbors and the extracted feature descriptors it can be checked if there is an error in the search. The closer are the distances the more probable is that it is an error case, because given the cardinality of the descriptors it is very improbable that they are similar. Then, a ratio threshold is used to discard these cases. If the distances ratio is lower than the threshold, the nearest neighbor and the extracted keypoint are considered as a match case and establish a 3D correspondence to the new keypoint. The ratio threshold used was 0.5, which is suggested by [3].

3.1.2. Classifier based

The keypoint matching problem can also be modeled as a classification one, as done by the Ferns algorithm [6], which uses a Naive Bayesian approach to solve it. In this approach, each keypoint is represented as a vector, where each coordinate represents the result of a simple binary test and each class represents all possible appearances of one keypoint of the model texture.

The preprocessing stage of this technique involves some steps. First, the model texture is searched for keypoints and a patch around each keypoint is extracted. From each patch, several images are generated by distorting it, so each class represents various possible views of it. For each generated image, the set of binary tests is applied in order to build the vector that will represent it.

3.2. Pose calculation

The problem of estimating the camera pose given n 2D-3D correspondences is called PnP (Perspective-n-Point) [19]. Several solutions have been proposed for the PnP problem in the Computer Vision and AR communities. In general they attempt to represent the n 3D points in camera coordinates trying to find their depths (which is the distance between the camera optical center C and the point M_i). In most cases this is done using the constraints given by the triangles formed from the 3D points and C. Then the pose is retrieved by the Euclidean motion (that is an affine transformation whose linear part is an orthogonal transformation) that aligns the coordinates. In this work, it was used the Robust Planar Pose (RPP) method [21] to solve the PnP problem. It is an iterative, accurate and fast solution that minimizes an error based on collinearity in the object space.

4. Web targeted implementation

As mentioned before, the keypoint based method was chosen to be used for performing some case studies regarding markerless AR for the Internet. Initially, it was adopted a distributed approach, where the tracking phase takes place at server-side. Based on the obtained results, it was implemented some versions where tracking is done at client-side. In both cases, it was used the RPP algorithm for pose calculation, due to its low execution time and the fact that it works well with planar objects, which was the kind of object used by the test applications.

Different keypoints were utilized, according to the features of the execution environment, as explained in the following subsections. In general, the Ferns classifier based technique is often faster than descriptor based techniques. However, in the case of client-side tracking, Ferns requires a much higher amount of precomputed data to be transferred to the client before tracking, which results in a slow loading application.

4.1. Server-side tracking

In the solution initially adopted, the client application was done in Flash, with the 3D rendering being done with Papervision3D. The client application communicates with Mammoth, which consists in a C++ open source Flash media server [7]. The architecture of Mammoth is depicted in Figure 5. The `rtmp_server` listens for incoming client connections using the RTMP protocol from Adobe [22]. It instantiates the `rtmp_connection` class when a connection is established. The `rtmp_connection` is responsible for parsing the RTMP message and delivering the request to the `service_manager` class, which handles the request and triggers the suitable `service` object. Each service has also a `stream` object, which decodes the media transferred from the client and invokes the processing to be done by the service. The developer needs to implement the `service` and `stream` classes for building the desired service.

In order to make possible the creation of AR systems utilizing this infrastructure, it was implemented a markerless tracking service for Mammoth using the keypoint based technique briefly described before. Two different matching approaches were used: a descriptor based one and a classifier based one. The descriptor based approach used an implementation of SIFT on GPU [4], since the server can have graphics hardware that is adequate to this processing, disobliging the client of having such hardware resources. An implementation of the Ferns classifier based technique was also utilized [6], because the precomputed data does not need to be sent to the client, as the processing is done on the server.

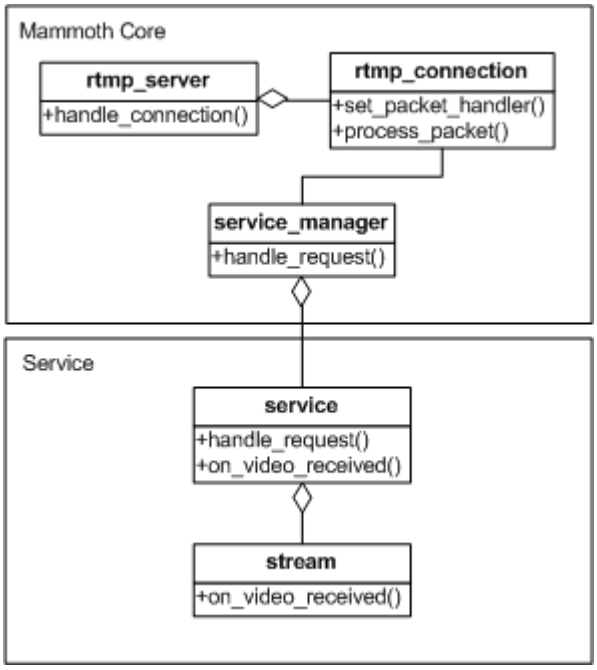

Figure 5. Mammoth architecture

In sum up, the application works in the following way: the video stream is captured by the Flash application at client-side and sent to the server using RTMP. The tracking service processes the video and communicates to the client if the object was detected. In case affirmative, the object pose is also sent, which

is used by the client to render the virtual information over the real image with the help of Papervision3D.

4.2. Client-side tracking

It was also adopted a standalone approach, where the markerless tracking occurs at client-side. Two standalone solutions were developed, each one using a different plug-in (OSAKit [8] and Microsoft Silverlight [9]). They are described next.

4.2.1. OSAKit

The OSAKit plug-in allows desktop applications to run without loss of performance inside most current browsers on the Windows platform. In order to publish the application on the Internet, some steps have to be fulfilled, as illustrated in Figure 6. First it is necessary to compact the application's executable file together with the dynamic link libraries (dlls) and other resources used in a `.zip` file. From the `.zip` file, a `.osa` file is generated by OSA Package Maker. This is the file that will be loaded by the web page in order to run the application.

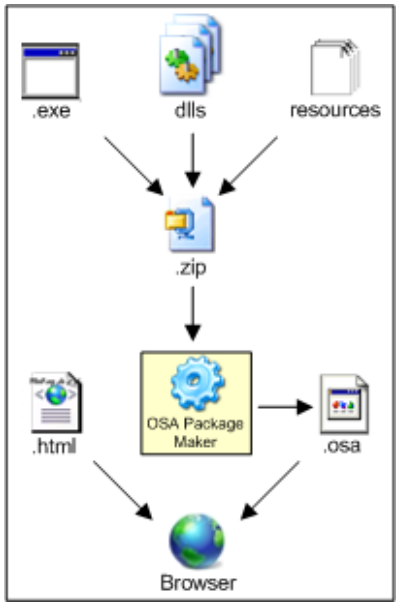

Figure 6. OSAKit package generation

Because the tracking will be done on client-side, there are no guaranties with respect to the presence of graphics hardware adequate for using SIFT on GPU. Due to this, it was utilized the SURF implementation [5], which runs on CPU and still has a good performance. Ferns was also used for performance comparison. The application dependencies were statically compiled, as a way to decrease the size of the data transferred to the client. 3D rendering was performed using the OpenGL library.

4.2.2. Silverlight

Microsoft Silverlight is a development platform that allows the creation of software applications for Web, desktop and mobile devices. Since the code is built against the .NET platform, the application is compatible with many browsers and operational systems [9]. In Silverlight 4 beta, which was recently released, webcam access was enabled, allowing the creation of AR systems.

The Silverlight Software Development Kit (SDK) takes as input the source code of the application, the dlls of the dependency libraries and any resources needed, and generates a `.xap` file that will be loaded by the user's browser, as shown in Figure 7.

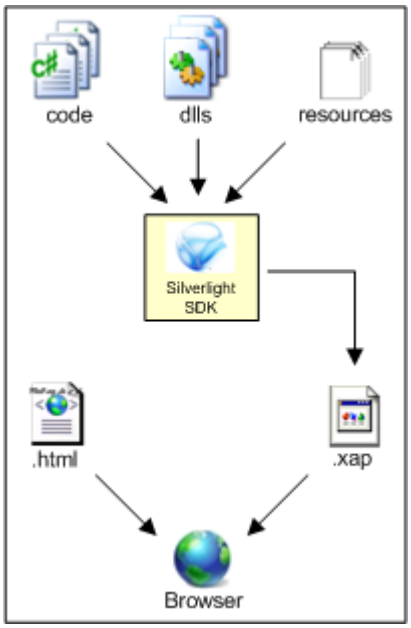

Figure 7. Silverlight application generation

The application built using Silverlight also makes use of client-side tracking, so the tracking method used should run efficiently on CPU in order to achieve the timing constraint. To meet this goal, it was used the Ferns technique, which also showed a good performance on desktop. The C++ implementation of Ferns cited in [6] was ported to the C# language subset supported by Silverlight. The original implementation depends on the OpenCV library [23]. Since it is not available as a full managed C# library, most of the functionalities that relied on OpenCV had to be adapted to use the Silverlight Application

Programming Interface (API). An exception was the code relative to linear algebra used by Ferns, which was adapted to use the dnAnalytics C# numerical library [25]. A portion of the dnAnalytics library relative to linear algebra was ported to Silverlight.

5. Results

The analysis of the developed solutions was done using a desktop computer with an Intel Core2 Quad 2.66 GHz CPU, 4 GB of RAM, a NVIDIA GeForce 9800 GX2 graphics board with 512 MB of memory and a screen resolution of 1440 x 900 pixels. The A4Tech ViewCam PK-635 camera was used, with a resolution of 320 x 240 pixels and a frame rate of 30 fps. The operating system is Microsoft Windows XP Professional x64 Edition SP2. The development tool utilized was Microsoft Visual Studio Professional Edition.

5.1. Server-side tracking

Figure 8 illustrates a web application that utilizes the described Mammoth based solution. A book cover is tracked and a music video is rendered over it.

Ferns presented a better performance than SIFT running on GPU, but the developed solution showed to be not scalable, causing a delay in the exhibition of the AR result when having more than five simultaneous users. This way, the adopted distributed approach turned out to be not adequate to web targeted applications, where hundreds of simultaneous users should be capable of using it. The developed solution can be more useful in applications for intranets with a limited number of simultaneous users. In this scenario, it is common that the available bandwidth is high and that the clients have low computational power, which are factors that promote the utilization of a distributed approach.

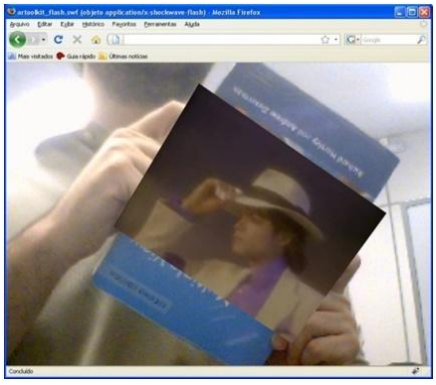

Figure 8. Markerless AR application for the web with server-side tracking

5.2. Client-side tracking

The following subsections describe the results obtained with the OSAKit and Silverlight plug-ins.

5.2.1. OSAKit

Figure 9 exemplifies the use of the described OSAKit based solution by an AR application where a virtual teapot is rendered over a book cover.

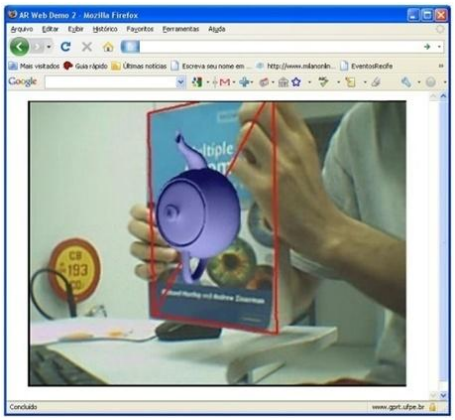

Figure 9. Markerless AR application for the web with client-side tracking using OSAKit

Table 1 compares the performance results obtained with OSAKit using SURF and Ferns for feature matching. As expected, choosing between SURF and Ferns is a tradeoff between frame rate and load time. Ferns outperformed SURF regarding frame rate, but its resulting `.osa` file size is much bigger (nearly 10 times), due to the size of the precomputed data. As a consequence, the average load time of the application when using Ferns is 10 minutes, while the load time when using SURF is just 1 minute, considering a broadband connection of 256 Kbps.

Table 1. Comparison of different feature matching methods using OSAKit

	SURF	**Ferns**
Average frame rate	20 fps	25 fps
`.osa` file size	1.8 MB	20 MB
Average load time (256 Kbps connection)	1 min	10 min

5.2.2. Silverlight

The application built using Microsoft Silverlight is illustrated in Figure 10. This time, no image was rendered over the detected book cover, but the text above the camera's image changed to display when the tracked object was detected or not.

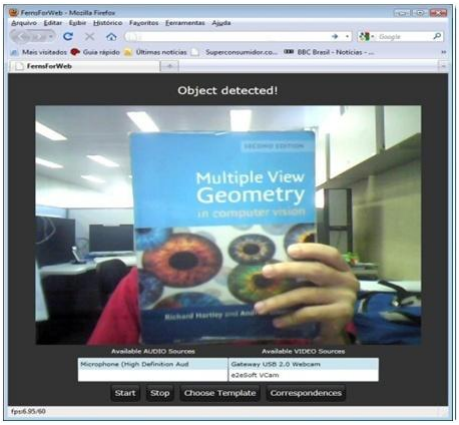

Figure 10. Markerless AR application for the web with client-side tracking using Silverlight

An average frame rate of 13 fps was obtained using this plug-in. The resulting `.xap` file size was 15 MB. Considering a broadband connection of 256 Kbps, the load time of the application was 8 minutes.

6. Conclusions and future work

In this work, some markerless tracking solutions for web targeted AR applications were developed, with the tracking processing being performed on the client or on the server. Client-side tracking showed to be more promising than the adopted server-side tracking approach, which presented scalability problems. The utilization of OSAKit had as its main advantage the fact that it uses the same application developed for the desktop platform and runs with the same performance, requiring little effort to make it available on the Internet. As disadvantages, it can be cited the restriction to the Windows operating system and the low diffusion of the OSAKit plug-in when compared to Adobe Flash, for example. The Silverlight solution shows a great potential, since it is platform independent and uses the C# language, making it easier to utilize the same software infrastructure for desktop and web. However, it still needs some performance improvements.

As future work, an investigation will be done in order to solve the scalability problem of the distributed approach. The Silverlight based solution will be extended to perform 3D augmentation and optimized to achieve higher frame rates. A descriptor based tracker such as SIFT or SURF will also be implemented in Silverlight for comparison. The performance results obtained using Silverlight will also be compared with a solution based on Adobe Flash, which is currently its most prominent competitor.

7. References

[1] World Premiere: The Augmented Reality Ad for MINI – Use Case by Metaio, http://www.metaio.com/fileadmin/homepage/Dokumente/English/Best_Practice_MINI.pdf, 2009.

[2] J. Lima, F. Simões, L. Figueiredo, V. Teichrieb, J. Kelner, and I. Santos, "Model Based 3D Tracking Techniques for Markerless Augmented Reality", *SVR*, SBC, Porto Alegre, 2009, pp. 37-47.

[3] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", *IJCV 60(2)*, Springer US, New York, 2004, pp. 91-110.

[4] C. Wu, "SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT)", http://cs.unc.edu/~ccwu/siftgpu, 2007.

[5] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features", *CVIU 110(3)*, Elsevier, New York, 2008, pp. 346-359.

[6] M. Ozuysal, M. Calonder, V. Lepetit and P. Fua, "Fast Keypoint Recognition using Random Ferns", *TPAMI 32(3)*, IEEE Computer Society, Washington, 2010, pp. 448-461.

[7] *Mammoth Server*, http://mammothserver.org, 2009.

[8] *OSAKitAbout*, http://www.osakit.com, 2009.

[9] *The Official Site of Silverlight*, http://www.microsoft.com/SILVERLIGHT, 2009.

[10] *FLARToolKit*, http://www.libspark.org/wiki/saqoosha/FLARToolKit/en, 2009.

[11] *Papervision3D*, http://code.google.com/p/papervision3d, 2009.

[12] *GE | Plug Into the Smart Grid*, http://ge.ecomagination.com/smartgrid/#/augmented_reality, 2009.

[13] *Metaio Unifeye Viewer 2.0*, http://www.metaio.com/products/viewer, 2009.

[14] *GE and Popular Science*, http://ge.ar-live.de, 2009.

[15] *Solutions – D'Fusion@Home*, http://www.t-immersion.com/en,do-it-at-home,34.html, 2009.

[16] *We Are Autobots – Transformers II: Revenge of the Fallen*, http://www.weareautobots.com, 2009.

[17] *Augmented Reality Demonstration - Natural Feature Tracking*, http://www.imagination.at/ARdemos/cabbage, 2009.

[18] *The Red Bulletin – Inside the World of Red Bull: Print 2.0*, http://en.redbulletin.com/print2.0, 2009.

[19] V. Lepetit, and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: a Survey", *Foundations and Trends in Computer Graphics and Vision 1(1)*, Now Publishers Inc., Hanover, 2004, pp. 1-89.

[20] J. Lima, F. Simões, L. Figueiredo, V. Teichrieb, and J. Kelner, "Online Monocular Markerless 3D Tracking for

Augmented Reality", Abordagens Práticas de Realidade Virtual e Aumentada, pp. 1-30.
[21] G. Schweighofer, A. Pinz, "Robust Pose Estimation from a Planar Target", *TPAMI 28(12)*, IEEE Computer Society, Los Alamitos, 2006, pp. 2024-2030.
[22] *Adobe - Real-Time Messaging Protocol (RTMP) specification*, http://www.adobe.com/devnet/rtmp, 2009.
[23] G. Bradski, and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly, Sebastopol, 2008.
[24] R. Hartley, and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, 2000.
[25] *dnAnalytics*, http://www.codeplex.com/dnAnalytics, 2010.