The Visual Computer (2021) 37:789–804
https://doi.org/10.1007/s00371-020-01830-8

ORIGINAL ARTICLE

A skyline-based approach for mobile augmented reality

Mehdi Ayadi[1,2] · Mihaela Scuturici[1] · Chokri Ben Amar[2] · Serge Miguet[1]

Published online: 6 March 2020

Abstract
This paper presents a skyline-based approach to enhance the visualization of a new construction project in augmented reality. We propose to process the video stream acquired with a mobile phone to register the real buildings with a 3D city model. We first combine the data acquired with the device's instruments to estimate a rough user's pose in the world coordinates system. Then, we use this estimated pose to generate a synthetic image of the user's view from which we calculate a *virtual skyline*. In parallel, we extract a *real skyline* from the real-time video stream. Finally, we match these real and virtual skylines to correct the user's pose (six degrees of freedom) and thus generate a more realistic augmented reality view. We evaluate the precision and the processing time of our approach using 2D and 3D registration algorithms, as well as with a novel double 2D strategy.

Keywords Mobile augmented reality · Image to geometry registration · 3D city models · Skyline matching · Urban landscape

1 Introduction

In recent years, the development of virtual reality and augmented reality technologies has automatically turned augmented reality to be the reference technology in terms of landscape simulation. Such systems appeared in the last decades with the development of *Smart Cities*. Furthermore, preserving a good visual landscape is important for inhabitants and can enhance their well being [17]. This can be done by allowing inhabitants to be involved in a city construction process, right from the design step. They can become acquainted with a building project by visualizing it as realistically as possible, integrating it into their living environment. Several augmented reality systems are already proposed, requiring innovative approaches and technologies. With augmented reality systems (AR), a user can achieve a more precise evaluation of the new surrounding environment with its new buildings and can also visualize the influence of new constructions on the global *skyline*.

On the other hand, devices available today in AR domain are mainly head-mounted displays (HMD) with augmented reality support. Such devices are expensive for individuals. Nowadays, new devices emerge at much more reasonable costs and users already own them (like smartphones or tablets).

The main challenge of our research is to propose an augmented reality system in urban context for smartphones. The idea is to merge real city images, acquired by the device's camera with virtual three-dimensional (3D) city model in order to position and insert a new building project.

In the literature, three main approaches exist: first, approaches relying on artificial visual features, namely *"fiducial markers"*. In our context, it is very constraining, sometimes impossible, to place specific markers visible everywhere in the city. They are more suitable for indoor environments, taking into account short distances and controlled conditions (luminosity, etc.). Second, solutions relying on device's embedded instruments. Such approaches are not very satisfying due to the presence of noise in outdoor environments, as shown in [9,10] or [25] (magnetic fields causing deviations of accelerometers and magnetic compass,

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s00371-020-01830-8) contains supplementary material, which is available to authorized users.

✉ Mehdi Ayadi
mehdi.ayadi@univ-lyon2.fr

Mihaela Scuturici
mihaela.scuturici@univ-lyon2.fr

Chokri Ben Amar
chokri.benamar@ieee.org

Serge Miguet
serge.miguet@univ-lyon2.fr

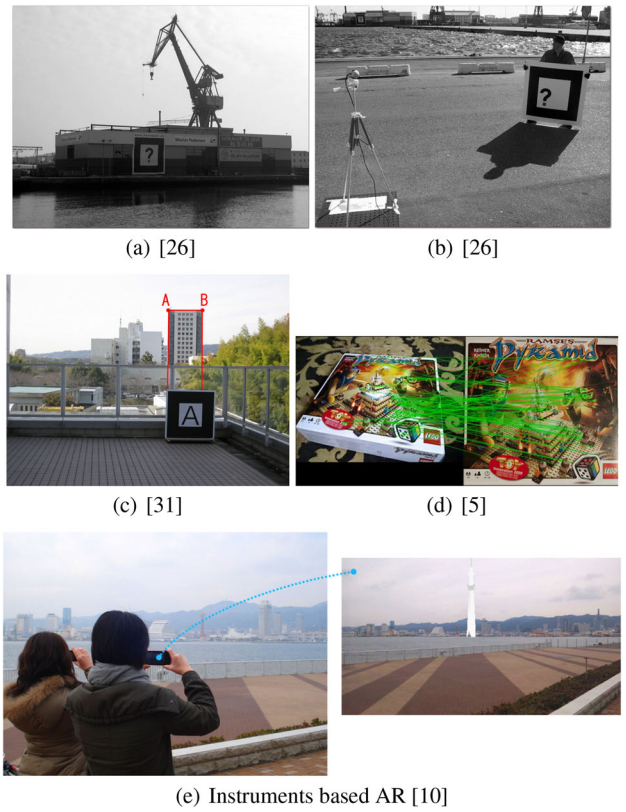
1 Université de Lyon, CNRS, Université Lumière Lyon 2, LIRIS, UMR 5205, F-69676 Bron Cedex, France

2 ENIS, REGIM-Lab: REsearch Groups in Intelligent Machines, University of Sfax, BP 1173, 3038 Sfax, Tunisia

GPS errors, etc.). Finally, with the growth of mobile devices performances, some solutions based on natural visual features have appeared [8] where the AR system rely on textured surfaces or specific points of interest.

In this paper, we propose a hybrid approach: we use the skyline (visual geometric feature) as a marker to register the real view acquired by the smartphone's camera and the virtual view of dynamically rendered 3D environment. For this, we use an estimated camera pose based on the smartphone's embedded instruments further refined by using the skyline. Our method was implemented and the proposed application was evaluated in the city of Lyon in France, with a basic hardware and software system, consisting in an iPhone 6 with integrated positioning system (GPS, barometer) and attitude sensors (gyroscope, accelerometer, magnetic compass). For software development, we used OpenGL-ES, Swift and *OpenCV* frameworks.

The remaining paper is organized as follows: In Sect. 2, we propose a literature review of *skyline extraction*, *augmented reality on mobile in urban context* and *skyline matching* approaches. In Sect. 3, we give an overview of the proposed approach. In Sect. 4, we briefly detail our skyline extraction method and in Sect. 5 our data fusion and pose estimation process. Then, in Sect. 6, we propose a refinement process allowing to correct the user's pose by using the skyline as a geometric feature. Finally, in Sect. 7 we present and discuss our results, propose conclusions and future works.



(a) [26] (b) [26]

(c) [31] (d) [5]

(e) Instruments based AR [10]

Fig. 1 Different AR approaches. **a**, **b** and **c** Artificial marker-based AR; **d** Natural markers AR-based; **e** sensor-based AR

2 Related works

2.1 Augmented reality approaches

Augmented reality was well defined in [3] and [20], as the technology that can superimpose the present real surrounding landscape acquired with device's camera and a 3D object. Much of the research was conducted in the field of landscape preservation as [21,22,25] or [10].

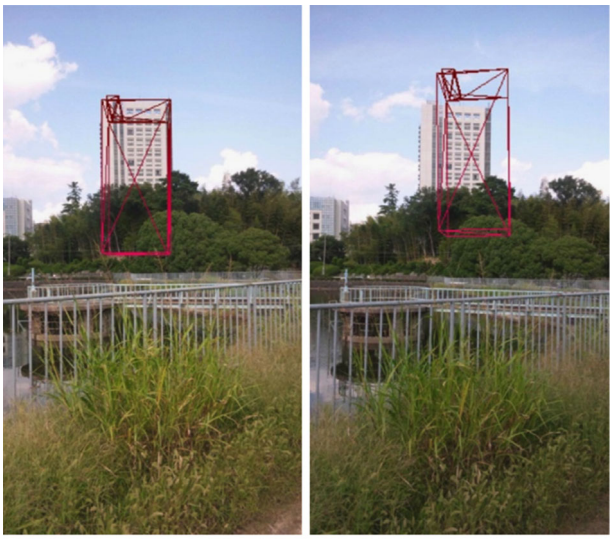
The common goal of all these approaches is to estimate the *camera pose*, i.e., the six degrees of freedom (3 in translations, 3 in rotations), assuming that the camera is calibrated, i.e., its intrinsic parameters are known (focal distance, dimensions and resolution of the camera's retina).

The first family of methods, relying on specific artificial markers, is appropriate for indoor environments, where placing a QR-Code or any other specific artificial marker is possible. Added that, most image acquisition conditions can be controlled, as luminosity, reflections, shadows, etc.

On the one hand, the advantages of such methods are mainly computing times and accuracy. In fact, recognizing a marker is a very simple task, due to very specific shape and colors that are not easily found in unconstrained environments. These traditional methods follow three steps: first, a binarization of the image using a threshold allows to obtain a binary edge map. Then, square contours are detected using, for example, the *Hough* transform [13]. Finally, corners are detected using for example the *Harris* detector.

On the other hand, a constraint has to be considered: artificial markers must always be present in the image, which considerably limits the user's mobility. Then, as in [26] or [31], the further the marker is placed, the larger it has to be, in order to be easily detected and to achieve high precision. We illustrate in Fig. 1a–c examples of such markers.

A second family of methods, the algorithms rely on device's embedded instruments to estimate the user's pose. During the last decades, to realize a highly precise registration, this kind of method required expensive hardware, as [6], or [29]. Using such methods made it unusable by the general public because it requires specific equipment that is not always available for most of users. Nowadays, hand-held devices (smartphone, tablets) include a multitude of sensors, allowing to estimate the user's pose (orientation, angular acceleration, GPS position, altitude, etc.). All of these capabilities allow to estimate the user's position (x, y, z), viewing direction (β) and orientation $(\alpha$ and $\gamma)$ and augment the real visualized scene with synthetic 2D or 3D objects according to user's pose.



(a) Ground truth [10] (b) Instruments reprojection [10]

Fig. 2 Instruments-based AR [10]

In the third family of methods, instead of having artificial markers (QR-Code), algorithms try to find natural markers in images. In fact, urban images are generally textured and contain a lot of information. These methods involve the extraction of feature points from images , such as *SIFT*, *SURF*, etc. These points are then matched with those found in reference images, as in [30]. We give an example of such methods in Fig. 1d.

The problem is that the instruments of a smartphone are not always reliable, especially in urban environment. In [16], the *C2B*[1] AR system allows to establish how construction processes could be improved with the use of new and innovative technologies, especially AR systems.

In [10], authors propose a landscape simulation system that position the user in the virtual world and achieves geometric consistency by merging the real image of a building with its 3D virtual wire model. The user's pose in virtual world is obtained by the fusion of all embedded instruments data for rotations and a map or GPS for lateral position. The authors evaluate the influence of each pose parameter ($Longitude$, $Latitude$, $Altitude$, $Roll$, $Pitch$, Yaw) on the system accuracy. Several experiments were conducted to evaluate the influence of each parameter on the system precision. In Fig. 2b, an illustration of instruments reprojection is compared to the ground truth, Fig. 2a.

In *City 3D-AR system* [9], authors present a system that allows architecture experts and urban planners to move around the city and project virtual 3D buildings on the real captured environment in a real-time video stream.

The problems in such methods are precision and the stability of the augmentation. In fact, the image is augmented with the 3D virtual building with a mismatch on both X and Y axis, also called *residual error* or *reprojection error*. Added to that, the augmentation stability is not satisfying and the added object appears to hover and float with the user's movements.

2.2 Skyline-based geolocalization

We oriented our research on the use of the skyline as a natural marker in urban context. Our goal is to propose a hybrid AR system for real-time video stream augmentation on mobile platforms using geometric features identified and extracted from the camera's images: the skyline. In this section, we propose an overview of skyline extraction and matching methods.

In the literature, we find several works studying the skyline in different disciplines. In Computer Science, a lot of interest has been shown in finding algorithms to extract the skyline from images. These approaches are clearly classified as image segmentation problems, using either traditional image processing techniques or more recent machine learning and deep learning approaches.

In [28] and [4], authors propose an approach based on SVM (Support Vector Machine), where the descriptors are essentially based on edges characteristics: color, location information and statistical features. This approach is well suited for images where just one skyline exists in an image. In fact, as in [1], several skylines at multiple depth levels may exist, as illustrated in Fig. 3, according to one of these two definitions:

1. The one-dimensional contour that represents the boundary between the sky and the ground objects [15];
2. The artificial horizon that a city's overall structure creates[2];

In [18], a neural network approach is used to extract the skyline. First, a neural network assigns to each pixel a score in [0..1]. Then, using a threshold, the algorithm keeps only pixels that have a score greater than a certain threshold.

Other works using the skyline for geo-spatial localization were carried out in [19], [24] or [23]. Authors propose a method to user's geolocalization in *urban canyons*, where skyscrapers block satellite signals causing GPS errors. Several tests were conducted in *Fujisawa* city, where an upward omni-directional camera is placed in the middle of the street delivering omni-directional images and the skyline forms a closed curve whose shape is a signature of the place where the picture is taken. Hence, sky pixels are located in the middle of the image. Skyline segments are computed from a coarse

[1] Pronounced "see to be".

[2] Wikipedia: https://en.wikipedia.org/wiki/Skyline.

(a) Natural skyline

(b) Urban skyline

Fig. 3 Multiple skylines on the horizon

3D city model and the extraction step relies on a graph cuts algorithm.

In [32], authors propose an approach for video / GIS registration to determine the camera's orientation from an overall synthetic panoramic skyline generated at the same position. The real skyline is compared to all its possible locations in the panoramic one (as a sliding window). Then, a cross-similarity function determines the user's orientation.

3 Proposed approach

Our 3D Augmented Reality system merges the real city images with a 3D modelization of the surrounding buildings using the skyline as a marker for user's pose refinement. In this section, we present our contributions. The flow of the developed system is illustrated in Fig. 4 and described as follows:

1. First, the smartphone's sensors are initialized in order to roughly estimate the user's pose. The system activates multiple sensors: the GPS, the accelerometer, the gyroscope, the magnetic compass and the barometer. We retrieve each data separately.
2. Then, a combination of all these values allows to obtain a rough estimation of the user's pose in the world's coordinates system:
   - user's position defined by longitude, latitude and altitude coordinates.
   - user's attitude defined by roll, pitch and yaw angles;

   Depending on the smartphone's capabilities, the user's altitude is either an absolute measure given by the DEM (Digital Elevation Model) or relative given by the smartphone's barometer by measuring the atmosphere pressure and then determining altitude. In the latter case, a calibration step is necessary to initiate the altitude at the beginning of the experiment assuming that the atmospheric condition do not vary during the experiments. Then, a request to a local database (embedded in smartphone) allows to find the 3D neighborhood's virtual model.
   - The caching mechanisms allow to ovoid the reload of geometric data that has already been downloaded;
   - Otherwise, a request (with the GPS position) is sent to the server, in order to retrieve 3D data.
3. By using the device's rendering support, a virtual image is generated from the 3D rendered data (buildings), the camera's intrinsic parameters and the previously estimated pose (example of a virtual image in Fig. 7b).
4. A live video stream is acquired by the device's embedded camera. Real images from the live video stream and virtual generated ones are separately processed with the same skyline extraction algorithm to, respectively, obtain a *real* and *a virtual* skyline. The user interacts with the app to set skyline extraction process parameters, as explained in Sect. 4.
5. A registration/matching process between the virtual and real skylines allows to refine either the user's pose or the user's attitude and thus improve the video's augmentation according to two criteria: precision (geometric consistency) and stability of the video augmentation. Note that this process is very memory consuming and thus is not always compatible with available computing performance on smartphone.
6. If the pose refinement is applied, a new user's pose is obtained. If the attitude refinement process is chosen, two main approaches are developed: 2D or 3D matching. The 3D matching is operated in the world's coordinate system to obtain a new user's orientation and viewing direction. If the 2D matching is chosen, a further step of 2D/3D conversion is applied to obtain the three rotational angles correcting the user's directions of observation.
7. Finally, occlusion treatments are applied to obtain only visible parts of the inserted 3D virtual building. The final augmented video shows more precision and stability than instrument-based one.

4 Skyline extraction

For the remainder of the paper, we assume the following assumptions:

- The sky is located in the upper part of the image.
- The skyline is defined, in our case, as the imaginary curve composed of edge points separating the ground objects and the sky;

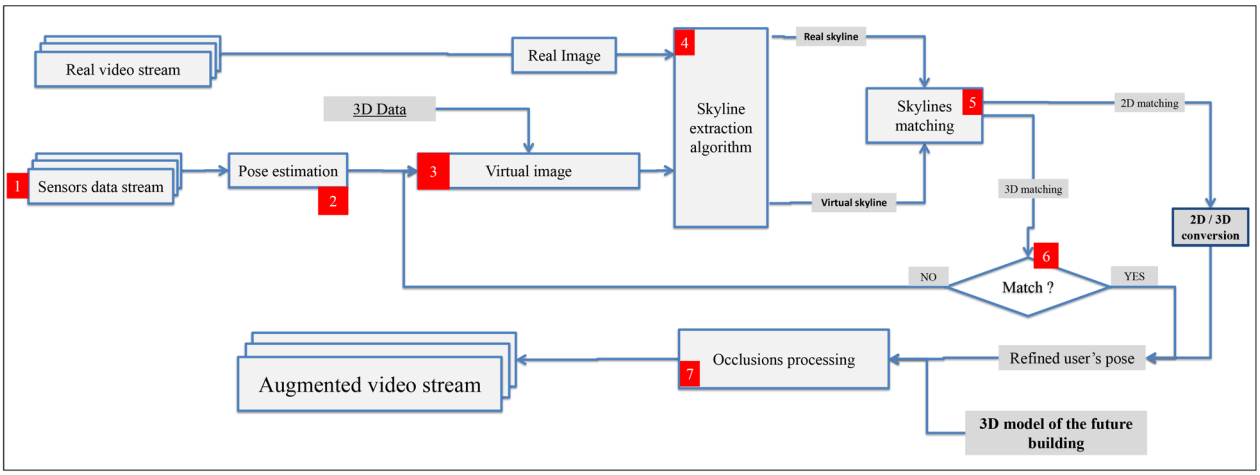


Fig. 4 System flow

In the following sub-sections, we present the skyline extraction steps: edge detection, upper envelope extraction and discontinuities connections.

4.1 Edge detection and upper envelope

First, the original image is transformed to grayscale color space and we detect edges by using the *Canny* edge detector [7], with its two parameters: *high* and low thresholds, set by the user depending on the version of the skyline he wants to obtain (see Fig. 3). This eliminates for instance, errors caused by clouds or cables that may appear in the image and disrupt the skyline detection. This edge detection step brings out only candidate pixels to be part of the skyline.

From this binary edge map, we extract an *upper envelope*: for each column of the image, from top to bottom, we keep the first point belonging to the edges. Once the entire image processed, we obtain an initial set of points called *upper envelope*, illustrated by the green segment pixels in Fig. 5a or b. This set of disconnected points certainly reflects the shape of the visualized scene but cannot be considered as the final skyline which is a continuous line. For this, a discontinuity connection step is needed.

4.2 Connecting discontinuities

From the previous edge detection and upper envelope step, an upper envelope is obtained from which we construct a multi-stage graph. Results are shown in Fig. 5. The upper envelope pixels have to be connected together. We represent the upper envelope pixels as graph nodes, thus obtaining a continuous curve representing the final skyline. This step connects the disconnected segments of the upper envelope using a path-finding optimization algorithm, as shown with red pixels in

(a) (b)

Fig. 5 Skyline extraction results

Fig. 5a, b. For more information on graph building approach, see [1].

The proposed skyline extraction algorithm is used to extract a *real* skyline from real images and a *virtual* skyline from the virtual city model, that have to be matched. To obtain the virtual skyline, we need a synthetic image corresponding to what the user virtually sees at a specific pose (position and orientation). In the following section, we detail our synthetic image generation process and the sensor fusion method for pose estimation.

5 Instrument-based augmented reality

The main goal of our hybrid AR system is 3D object placement in real video stream. We qualify it *hybrid* due to the use of:

– First, smartphone's embedded instruments to initialize the rendering process;

– A vision-based algorithm to refine the user's pose and correct and stabilize video augmentation;

In the following, we detail the 3D data used in our test campaigns (Sect. 5.1), our rendering process, the camera model construction and our pose estimation process (Sect. 5.2)

5.1 Dataset

Our data is the 3D city model of *Lyon's city in France*, which is an open source model provided by the *Metropolitan Lyon services*, representing about 550 square kilometers in *CityGML* format.

As of today, many cities around the world own their virtual double [11]: NewYork,[3] Brussels[4] or Lyon,[5] and become more and more available to users. For *Lyon*'s city model, the *CityGML* format is used [6]. However, whole city data are about 400 GB, which obviously cannot be stored on a mobile device.

To overcome this problem, we set up a client-server mechanism allowing the user to retrieve the 3D data of the neighborhood, in case this data is not present in the phone's cache. In fact, a mechanism of cache management is implemented, to verify that the data are not yet present in smartphone's local database.

5.2 Rendering process and pose estimation

The rendering process is essentially based on the camera model construction and sensor's data retrieving and fusion. Before proceeding to the pose estimation step, we go through several intermediate stages. First, the real camera is calibrated using *OpenCV* framework to obtain its intrinsic parameters. Then, the camera model is constructed to be able to render 3D object on the screen. Finally, different *OpenGL-ES* buffers are supplied with 3D data: normal's buffer, textures buffer, geometry's buffers and texture images. Using various *OpenGL-ES* directives and contexts (for rendering), specific content is drawn and different images are built: virtual or augmented ones (see Fig. 13a–c, etc.). Note that the texture data are not explicitly used by our matching algorithm, but rather visualization and augmentation purposes.

Basically, the pinhole camera model allows to project 3D object coordinates in the WCS (World Coordinate System) on the smartphone screen in the ICS (Image Coordinate System), obtaining its 2D coordinates [14]. The projection matrix can then be written as follows:

$P = K * R * T$, where K is the camera intrinsic parameter matrix, R is the rotation matrix and T is the translation matrix.

The goal here is to position a virtual camera in the 3D city's model with the same user's attitude. The user's attitude is estimated with the smartphone's instruments and described as:

– A position (x, y, z);
– A sight direction (or viewing direction: Roll (β));
– An orientation (pitch (α) and yaw (γ));
– A moment (real-time augmentation).

The user's position (x, y, z) is acquired with the smartphone's GPS feature, where (x, y) is directly acquired by the GPS (longitude, latitude) in the *WGS84* referential, that will be later converted in *RGF93-CC46* referential.[7] The *altitude* (z) is retrieved from a DEM or from the barometer.

We go over all projection and rotation matrices definitions on each axis ($R_X(\alpha)$, $R_Y(\beta)$ and $R_Z(\gamma)$) (detailed in the *supplementary material*). We illustrate in Fig. 6 the smartphone's axis, where the *Roll* angle (β) is directly retrieved from the magnetic compass. The *Yaw* (γ) and *Pitch* α angles are determined using the *tri-axial accelerometer*. The output of the accelerometer can be modeled as follows:

$$\vec{a} = \begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix}$$

where: a_X, a_Y, a_Z are acceleration along the X, Y and Z axis.

We consider the user's attitude to be unconstrained. In this case, the three rotations are performed successively obtaining the user's posture in real-time. Then the $R_X(\alpha)$, $R_Y(\beta)$

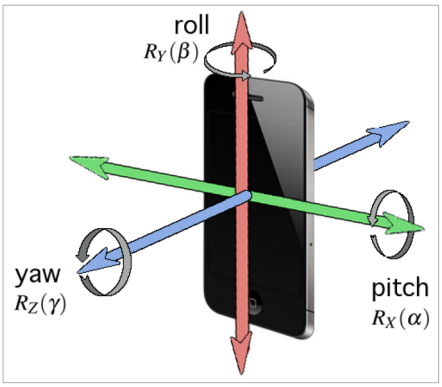


Fig. 6 Smartphone's axis

[3] https://www1.nyc.gov/site/doitt/initiatives/3d-building.page, 2019.

[4] https://cirb.brussels/fr/nos-solutions/urbis-solutions/urbis-data/urbis-adm-3d , 2019.

[5] http://data.grandlyon.com, 2019.

[6] http://www.opengeospatial.org/standards/citygml, 2019.

[7] https://proj4.org/about.html.

(a) Real image (b) Virtual image (c) Skylines

Fig. 7 Instrument-based rendering. Real skyline in red; Virtual skyline in green; $A_{real} = (154, 184)$, $A_{virtual} = (259, 176)$; Error in pixel: $(\Delta_x, \Delta_y) = (-95, 8)$; Error in degree: (8.54, 0.67)

and $R_Z(\gamma)$ matrices have to be multiplied together. A choice has then to be made for the multiplications order, because matrix multiplication is not commutative. In fact, from the six possibilities, only R_{XYZ} and R_{YXZ} are exploitable solutions (depend only on α and γ) and all others are unsolvable. We give in the *supplementary material* all the combinations results. This gives us the following result:

$$\alpha = \tan^{-1}\left(\frac{-a_X}{\sqrt{a_Y^2 + a_Z^2}}\right) \quad \gamma = \tan^{-1}\left(\frac{a_Y}{a_Z}\right)$$

where: γ (yaw) and α (pitch) are, respectively, the rotational angles around the Z and X axis.

At this step, the camera model is built with its intrinsic and extrinsic parameters. The city visualization based on instruments augmentation can then be done in real-time (see video: *InstrumentalAR.mov* in the *supplementary material*).

Limitations:

We illustrate in Fig. 7a the real image acquired by the smartphone's camera, and in Fig. 7b the corresponding virtual image generated from 3D model and the *user's estimated pose using only instruments* (GPS, accelerometer, compass).

From each of these images, we extract the real and virtual skylines.

Due to instruments measuring errors, we notice that the real image (Fig. 7a) is *different* from the virtual one: the user's pose is not 100% accurate and thus we notice errors in terms of geometric consistency. We annotate the same point A on the two images to better illustrate this error.

To overcome these drawbacks, we propose to add precision to the video augmentation process and increase system precision by using the skyline as a marker for user's pose refinement.

6 Skyline matching

In this paper, we situate our contributions in the registration methods based on geometric features. The proposed approach is described as a complementary process, where the skyline matching step is engaged under the assumption that significant sky regions exist in the real image and a 3D model of the visualized scene exists. However, the augmented reality system relies on instruments to augment the scene.

We allow the user to place himself at any position and any orientation (six degree of freedom), as explained in Sect. 6.2. In the following, we present the matching process as an extension to [2], where different similarity metrics were proposed.

This process is divided in three main steps:

- First a skyline *pre-processing* step to eliminate outliers in the real skyline, that might come from the extraction process (Sect. 4).
- Then, we apply a polygonal approximation of the skylines to keep only significant geometric features and avoid unnecessary details hampering the matching process.
- Finally, an optimization algorithm search for the best user's pose while minimizing the *distance* between the virtual and real skylines.

The typical scenario is that the user is moving while visualizing the augmented video stream with the future building. We investigate two main approaches that are treated as minimization problems according to different variables.

The first approach discussed in Sect. 6.2 refines the user's position and orientation. We try to find the real user's position and orientation while having a first rough estimation from the instruments. The minimization algorithm's variables are: $Longitude$, $Latitude$, $Altitude$, $Roll$, $Pitch$ and Yaw.

The second approach is discussed in Sect. 6.3, where the user is fixed at an *a priori* known position, either directly

retrieved by the GPS or given by the user (using a map centered on his actual position). In this approach, our system refines only the user's attitude (three rotational angles). This time, the minimization algorithm's variables are: $Roll$, $Pitch$ and Yaw.

6.1 Matching process

6.1.1 Skyline pre-processing

We noticed, in many examples of our database, that extracted skylines contain a lot of straight lines artificially created during the skyline extraction step, caused by noises in images and skyline's extraction failures. Added to that, the original skyline's size is proportional to image's width. Initializing the matching process with such size compromise the process, the real-time constraint is no longer respected. To bypass this constraint, we add two pre-processing steps: skyline simplification and polygonal approximation. This allows to significantly reduce the skyline's size within a few dozen points, while reflecting the shape of the scene and keeping only important geometric features, while avoiding unnecessary geometric details.

The skyline simplification step is detailed in Algorithm 1, where:

- "input" is the original skyline vector;
- "output" is the resulting vector;
- "precision" is a parameter reflecting the magnitude of allowed vertical jump between two consecutive pixels.

Algorithm 1 Aberrant pixel removal

```
for i = 1, to sizeof(input) do
  if |input[i+1].y + input[i-1].y - 2*input[i].y| < precision
  then
    output.append(input[i])
  end if
end for
```

Once the outliers removed, we apply a polygonal approximation on the skyline obtaining a new curve S_{approx} with less vertices, where the distance between the original skyline S_{orig} and S_{approx} is less than the *precision* parameter. The polygonal approximation used in this paper is based on the Douglas–Peucker algorithm which implementation is inspired from OpenCV framework.[8]

The purpose of this polygonal approximation is to decrease the size of the skyline vector. The idea here is to obtain a vector whose size is proportional to the complexity of the visualized scene rather than to the resolution of the image. In other words, we only keep from the skyline a description of the most robust geometric features and eliminate the very fine details that could hamper the registration process.

Note that all illustrations in Fig. 10 are the polygonal approximations of both real and virtual skylines.

As it will be discussed in Sect. 7.1, all images, videos and intermediate results are available under *CC-BY creative commons license*. An example of images of skylines before and after simplification process is given in Fig. 8. We present, in Table 2 of the supplementary material, a comparison of vectors size before and after this pre-processing process for different values of the *precision* parameter.

The registration process is impacted when the precision parameter changes. If the precision parameter is zero, no simplification is performed and the original real skyline is kept as is. In this case, aberrant pixels, small architectural details or small indentations in the skyline are taken into consideration and the real skyline will differ a lot from the virtual one : the registration process then fails and high-frequency oscillations in the augmenting model are observed. Otherwise (precision parameter is not null), we get a skyline in which the aberrations are deleted (isolated pixels, etc.). Thus, the registration process behaves well as the two skylines are quite similar and the user's pose refinement succeed.

6.1.2 Minimization algorithm

The minimization algorithm used here is the downhill simplex algorithm.[9] We have to define a comparison metric that determines the distance between the real skyline S_1 and the virtual one S_2. This metric should be minimum when the skyline of the estimated camera pose yields the skyline of the real user's pose. In this paper, we focus on different strategies using the same similarity distance, L_1, which represents the area between the two curves S_1 and S_2, as shown in Equation 1. Several distance functions were tested in [2].

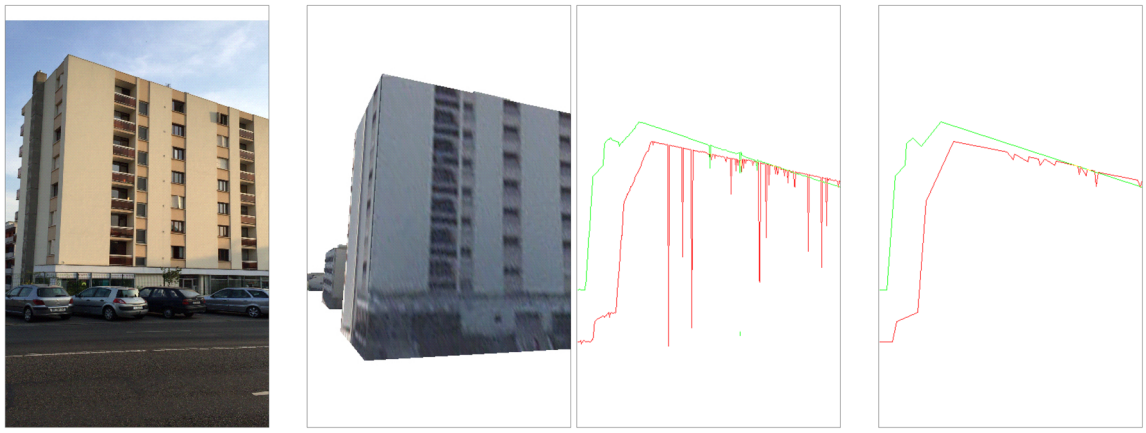
$$L_1(S_1, S_2) = \int_x \mid (S_1(x) - S_2(x)) \mid \mathrm{d}x \tag{1}$$

Then, we adopt a strategy where the algorithm's variables are dynamically changed along the process. Then we have multiple and different steps for each variable in the parameter space to prevent the minimization algorithm from being stuck in local minimas. Finally, steps decrease along the whole process, thus converging to the global minimum. For each step, the whole process is restarted from the previous solution until convergence.

In the following, we detail each investigated approach: refining the user's pose or refining the user's attitude, both in terms of precision and in terms of computation time.

[8] https://github.com/opencv/opencv/blob/master/samples/cpp/contours2.cpp.

[9] http://en.wikipedia.org/wiki/Ramer-DouglasPeuckerAlgorithm.



(a) Real image (b) Virtual image (c) Skylines before simplification (d) Skylines after simplification

Fig. 8 Skyline simplification: image 17

Table 1 Results table

Video	Error (pixels)				Processing time (ms)							
	Average				iPhone 6				iPhone 7 Plus			
	instr	2D	2×2D	3D	instr	2D	2×2D	3D	instr	2D	2×2D	3D
Video 1	49.62	24.88	6.34	4.24	10.33	53.43	99.74	5083	8.90	35.09	63.48	1860
Video 2	26.55	13.19	5.85	5.22	10.21	58.72	103.76	5705	8.89	29.33	62.87	2125
Video 3	17.37	11.32	5.23	4.89	11.32	58.96	98.11	5178	9.79	37.18	62.95	1940
Video 4	28.87	8.33	4.92	4.23	10.11	59.14	96.12	4932	9.07	37.21	63.13	1880

6.2 Pose refinement

This first approach is to refine the six degrees of freedom. At the beginning, the user localization is based on the gps signal and the orientation on device's instruments. To correct the user's pose (position and attitude), the optimization algorithm has to explore a parameter space of six dimensions: *Longitude*, *Latitude*, *Altitude*, *Roll*, *Pitch* and *Yaw*.

Our first experiments show that the most unstable parameter is altitude based on the GPS, with sudden 30 m changes, giving a starting point too far from the real one. Since our experiments were performed on a quite lat are, we decided to fix the altitude and to let only the five other parameters vary.

As explained in Sect. 3, this approach cannot be established in real-time. In fact, when exploring the parameters space, the virtual image has to be regenerated for each new user's pose. This process is re-iterated until the algorithm converges to its global minimum, considered as the real user's pose.

For instance, we illustrate in Fig. 9, the parameter space for image 200 of video 5 in our database. Corresponding real, instrument-based and skyline-based images are given in the associated supplemental material. We explore this domain exhaustively, with a step of 50 cm, on each *Longitude* and *Latitude* axis. For each position, we search for the best user's viewing direction and orientation that minimizes the measure between the real and the virtual skyline. Note that the domain is regular and thus the algorithm is able to converge to the real user's pose.

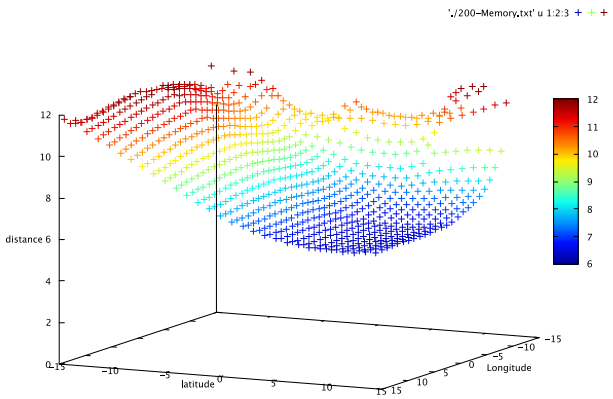
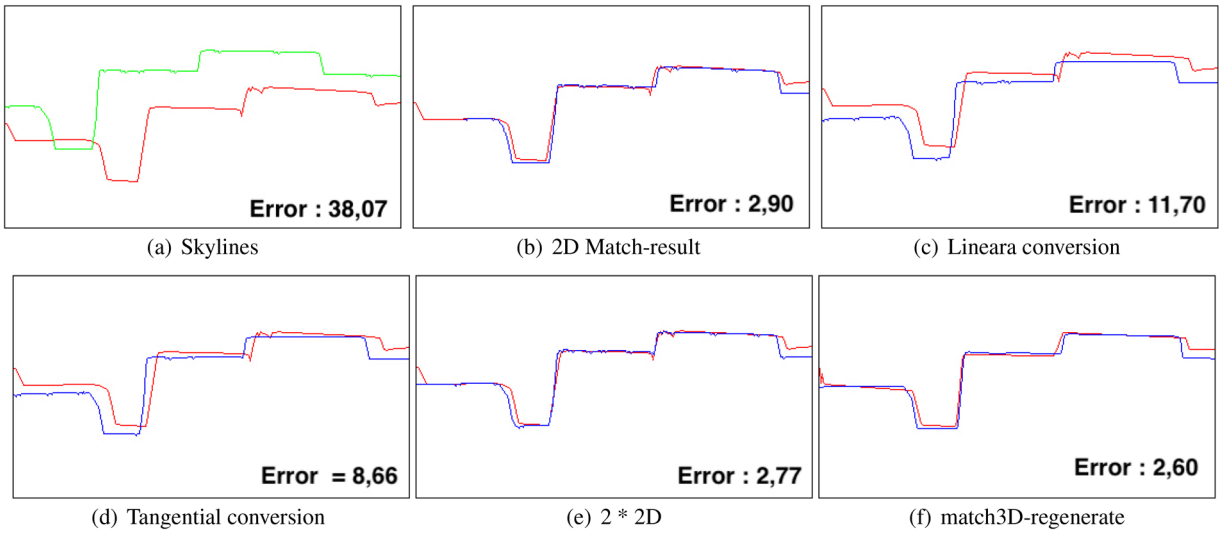


Fig. 9 Exhaustive domain exploration: image 200 of video 5

6.3 Attitude refinement

In the following, we detail the user's attitude refinement approach with its two variants:



(a) Skylines (b) 2D Match-result (c) Lineara conversion

(d) Tangential conversion (e) 2 * 2D (f) match3D-regenerate

Fig. 10 2D and 3D matching approaches (error in pixels)—video 1. In red: real skyline; in green: instrument-based virtual skyline; in blue: virtual skyline after matching process

1. A matching process in the world's coordinates system, called *3D Matching*, with three variables: α, β and γ corresponding to the user's viewing direction and orientation. The optimization algorithm tries to find the best triplet so that the virtual skyline looks the most like the real one. This approach will be presented in Sect. 6.3.1.
2. A matching process in the image coordinates system, called *2D Matching*. The principle of this method is to *move* or *translate* the virtual skyline in the image plane along X or Y axis and *rotate* it around image's center. The final solution is then ΔX, ΔY pixels and $\Delta\theta$ radians. Knowing the camera's intrinsic parameters and the device's screen size, we can convert this transformation (ΔX, ΔY, $\Delta\theta$) into three rotational angles ($\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$) that correct the user's viewing direction and orientation. This conversion is an approximation which is valid only for small angles for which the transformation induced by rotations of the viewing direction can be assimilated to translations.

6.3.1 3D matching

The minimization algorithm explores the parameter space to find the best user's orientation (triplet of α, β and γ) while minimizing the distance between the two skylines. In this approach, for each iteration of the optimization algorithm, the virtual image corresponding to that pose is regenerated. This regeneration takes about 10 ms on an iPhone 6 or 8 ms on an iPhone 7 Plus. This approach is accurate but is very time-consuming. As discussed in Sect. 7, the algorithm finds the best solution (P_{final}) in about 100 iterations in average. The entire process of skylines extraction, pre-processing and matching takes then 1.8 seconds on an *iPhone 7 plus*, which is suitable for still images or photo-realistic augmentation but not for real-time video augmentation. We illustrate in Fig. 10f the results of this algorithm, where the virtual skyline is *perfectly* aligned with the real one and the distance between the two skylines is 2, 77 pixels. This approach which is faster but less precise will be described in Sect. 6.3.2.

6.3.2 2D matching

In this approach, the optimization algorithm tries to find the best:

- Translation of ΔX along the X-axis;
- Translation of ΔY along the Y-axis;
- Rotation of $\Delta\theta$ around image's center;

As in [12], small *pitch* and *yaw* rotations can be considered as translations in the image plane. The Roll angle is a rotation around the image's center. This approximation is available under two assumptions:

- We consider that instrument's data are imprecise but not totally aberrant. Thus, the rotational error is not very important as compared to the camera's field of view (about 34 degrees in portrait mode).
  For instance, the magnetic compass gives an estimation of the magnetic north with a precision of about 15°. For instance, in video 1 of our database, the horizontal average error is 5.37° and the maximum is 11.78°.
- The observer's distance to the visualized objects (d) has to be significantly larger than the positioning error (ϵ): $d >> \epsilon$. In fact, this assumption is verified since we have:

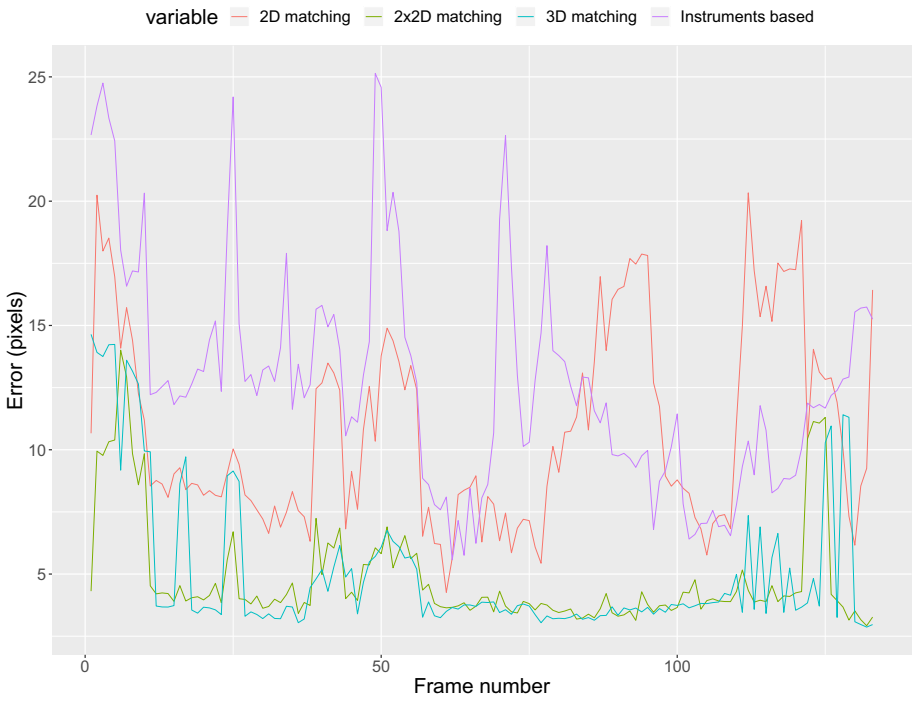


Fig. 11 Comparison of different approaches (video 3)

– The gps positioning error is about 5–10 m in urban context (except in urban canyons cases where it can be larger)
– The use case of our AR application is to assess the effect of a new construction projet on the urban landscape and thus, the user has to position himself at a certain distance of the visualized objects.

We illustrate in Fig. 7, an example where the virtual skyline (Fig. 7b) has to be moved by 95 pixels along the X-axis, 8 pixels along Y-axis and rotated by 2.12° around Z-axis.

The 2D matching process starts with the two skylines of Fig. 10a to obtain a *sub-skyline* in Fig. 10b, where the skyline was moved by 36 pixels along X-axis, 39 pixel along Y-axis and rotated by 1.16° around image center. Note that we do not obtain a complete skyline, but only the part that matches the virtual to the real one. The distance between the real skyline and this sub-skyline is about 2.90 pixels. We visually observe that the skylines are almost perfectly aligned with little differences due to geometric imprecision of the 3D virtual model.

From this 2D transformation, the 2D / 3D conversion process (step 8 in flowchart of Fig. 4) consists in finding the three rotation angles (α, β and γ) allowing to generate the virtual image corresponding to the refined user's pose. A naive approach would be to operate a linear conversion on these translation vectors obtaining the result as shown in Fig. 10c, where the distance between the two skylines is 11.7 pixels. This approximation can be defined as follows:

$$\beta = \frac{\Delta x}{w} \cdot \mathrm{FOV}_x.$$

where w is the image's width, FOV_x is the field of view on the X axis; Δx the translation vector along the image X-axis; β the rotation angle around Y-axis in world coordinate system.

A more rigorous approach is to use the camera's field of view (FOV) and screen's size. Results are shown in Fig. 10d, where the distance between the two skylines is 8,66 pixels. This approximation is defined as follows: $\beta = \arctan(\frac{\Delta x}{w \cdot \tan(FOV_x)})$.

On the one hand, the 3D matching approach is very precise but cannot be computed in real-time. On the other hand, the 2D approach is real-time but its precision is lower. For this, we propose the double 2D matching process denoted 2×2D (see Fig. 10e).

6.3.3 Double 2D matching

We investigate a double 2D matching process (**2 × 2D**), where a first 2D matching allows to estimate the user's pose (P1), from which a virtual image is generated. Then, the process is re-iterated starting from P1, to obtain the final user's refined pose (P_{final}). We notice, compared to Fig. 10c, d, that this strategy gives better results, as illustrated in Fig. 10e where the distance between the two skylines is 2,77. These results are comparable to the 3D matching results (discussed above) of Fig. 10f.

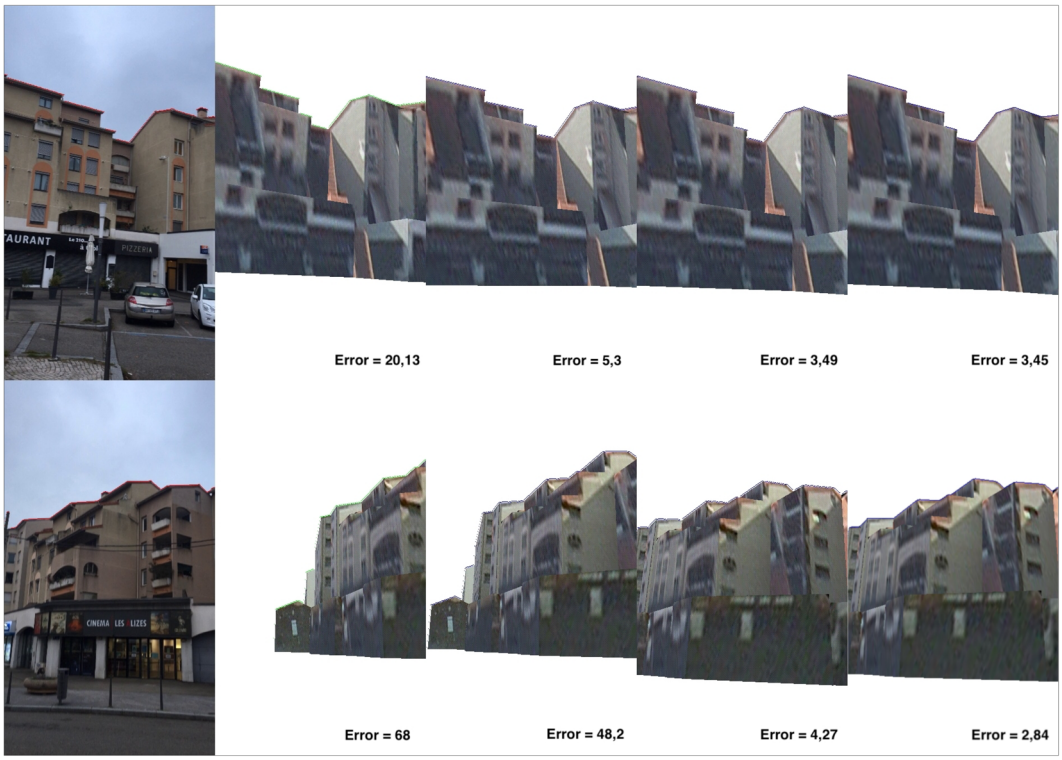


Fig. 12 Matching results. Line 1: frame 48 video 1; Line 2: frame 17 video 4

7 Results and discussion

As mentioned in the introduction, the first version of our system was realized on iPhone 6 that includes support for high performance 2D and 3D graphics based on the OpenGL-ES (Embedded Systems) API. For software developments, we used iOS-SDK and XCode. For image processing algorithms, we developed our proper algorithms for skyline extraction based on the *OpenCV* framework. For AR visualization, two *OpenGL-ES context* panels are used to overlay multiple views in the same time in full screen mode: in bottom, we depict data acquired from live-video stream and above the 3D model data. We deployed our system on iPhone 6 and iPhone 7 Plus and tested it in the city of *Lyon*, France. In the following, we present the dataset constructed during this research project and our results.

7.1 Matching database

In this subsection, we detail the constructed databases during this research. This database was used to evaluate our system performances.

In fact, given that no dataset exists with the information required by our system which are, the real image with its *GPS coordinates* and *an estimation of camera's orientations parameters*, we created our proper database which is publicly available.[10] To create this dataset, we developed a smartphone application allowing to record a real-time video of the visualized scene and meanwhile records instrument's data. From retrieved data, the instrument's pose P_{instr} is calculated as detailed in Sect. 5.2. Then, once our registration process applied, the user's refined pose is calculated and the final distance between the two skylines Err_{reg} is compared to the initial one Err_{instr}. This database is available under CC-BY creative commons license and allows further works in different communities.

Then, for each of the acquired videos, we associate instrument-based synthetic images, the registered images using our 2D or 3D matching process and the final augmented videos for each of the proposed approaches. Consequently, we created a database containing about 2000 images.

Finally, for the video augmentation and demonstration, we propose to use the *Crayon Tower* 3D model from *Lyon's business district*, and add it to our testing area (neighborhood) where database was created. Using the actual 3D city model, the real image and the future construction building, we calculated occlusions and augmented the scene with only the visible part of the 3D building model. Examples are shown in Fig. 13f, e.

[10] https://perso.liris.cnrs.fr/mayadi/LyonGeoTagged.

(a) Instruments (b) Instruments (c) Instruments

(d) Match3D (e) Match3D (f) Match3D

Fig. 13 Augmented images with Crayon tower

7.2 Results

We present in Fig. 11 the results of our registration process for video 3 in our database, where abscissa value corresponds to the frame number and the ordinate is the L_1 measure between the two skylines. We notice that, for each frame, the instrument-based approach curve has an important error as compared to the registered ones. We calculate for each video the average error of each approach: instrument base approach, 2D, 2×2D and 3D matching (Table 1). As discussed above, the 3D matching approach gives the best results as compared to the 2D or 2×2D ones.

The first column (left) in Fig. 12 corresponds to the real images. The other columns correspond to the virtual images generated using the our approaches described above. We note that the error between the system projection error decreases from instrument-based approach to the 3D matching one.

Figure 13a–c presents examples where the image is augmented using the instrument-based approach. We note that the tower is inserted in a wrong location. Then, respectively, in Fig. 13d–f, the tower is registered using the 3D matching process in a much more precise location.

For the real-time constraint, the 2D matching process is compatible on both platforms (iPhone 6 or iPhone 7 Plus). In fact, the whole process with skyline extraction, simplification, polygonal approximation and registration step can be operated in real time. On the contrary, the 3D matching process cannot be established in real time. A good compromise between processing time and video augmentation precision is the *2×2D* matching approach, where we can obtain a precision that is comparable to the 3D matching process while respecting the real-time constraint.

Table 1 presents the average processing time in milliseconds for the whole process for each approach.

To evaluate our system, we compared our results to the instrument-based approach. This comparison allows us to clearly notice the advantages of the skyline-based registration in terms of robustness and precision. To the best of our knowledge, no other methods combined, in real time and on a general public device (smartphone), both "*a sensors-based*"

and "*a vision-based*" system at the same time. Moreover, our approach is robust to observation conditions especially when the real and virtual skylines present some differences due to: missing data, jagged skyline, too much vegetation, etc.

8 Conclusion and future works

We presented a hybrid augmented reality system that registers the real-time video stream with 3D models of urban scenes. This hybrid approach allows to first estimate the user's pose using the smartphone's embedded instruments, which is not accurate. We combine this sensor-based system with a vision-based one, based on geometric characteristics of the scene. These geometric characteristics are skyline pixels. We then show that the skyline can act as a marker for an augmented reality application in urban context. We proposed a parametric skyline extraction algorithm, followed by a skyline matching process with different approaches. Our system is fully functional for real-time video augmentation with algorithms running efficiently on an *iOS* platform. We proposed multiple expandable databases of real and virtual skylines, 3D models and meta-data taken in city of *Lyon*, *France*. All database are publicly available.

The experiments made in this paper consist in different cases where vegetation is either present or not in the real environment. If that is the case, the matching process fails. We currently investigate two solutions: the first is based on deep learning. A neural network is pre-trained on our dataset and used in real time on smartphone to extract the skyline with a better precision than edge-based method. The second is to consider a discontinuous skyline where the parts of the skyline corresponding to vegetation are not taken into account in the registration process. For this, we have the opportunity to use the results of thesis work of [27] to characterize the visualized scene using the skyline. Indeed, this approach allows to say which part of the skyline is drawn by a natural or artificial element. Other perspectives are to find a solution where the pose refinement process could be optimized with different strategies that can be established in real time on smartphone. To finish with, we investigate a spatio-temporal strategy where we initialize the optimization algorithm at previous frame's pose (P_{t-1}) rather than processing all frames independently.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ayadi, M., Suta, L., Scuturici, M., Miguet, S., Ben Amar, C.: A parametric algorithm for skyline extraction. In: Blanc-Talon, J., Distante, C., Philips, W., Popescu, D., Scheunders, P. (eds.) Advanced Concepts for Intelligent Vision Systems, pp. 604–615. Springer, Cham (2016)
2. Ayadi, M., Valque, L., Miguet, S., Scuturici, M., Ben Amar, C.: The skyline as a marker for augmented reality in urban context. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Porikli, F., Skaff, S., Entezari, A., Min, J., Iwai, D., Sadagic, A., Scheidegger, C., Isenberg, T. (eds.) Advances in Visual Computing. Springer, Berlin (2018)
3. Azuma, R.T.: A survey of augmented reality. Presence: Teleoper. Virtual Environ **6**(4), 355–385 (1997)
4. Baatz, G., Saurer, O., Köser, K., Pollefeys, M.: Large scale visual geo-localization of images in mountainous terrain. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) Computer Vision—ECCV 2012, pp. 517–530. Springer, Berlin (2012)
5. Baggio, D.L., Emami, S., Escriva, D.M., Ievgen, K., Mahmood, N., Saragih, J., Shilkrot, R.: Mastering OpenCV with Practical Computer Vision Projects. Packt Publishing, Limited. Recommended: advanced OpenCV project support/examples inc. iOS and Android examples (2012)
6. Behzadan, A.H., Timm, B.W., Kamat, V.R.: General-purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering. Adv. Eng. Inform. **22**(1), 90–105 (2008)
7. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI–8**(6), 679–698 (1986). https://doi.org/10.1109/TPAMI.1986.4767851
8. Carozza, L., Tingdahl, D., Bosché, F., van Gool, L.: Markerless vision-based augmented reality for urban planning. Comput. Aided Civ. Infrastruct. Eng. **29**(1), 2–17 (2014). https://doi.org/10.1111/j.1467-8667.2012.00798.x
9. Cirulis, A., Brigmanis, K.B.: 3D outdoor augmented reality for architecture and urban planning. Procedia Comput. Sci. **25**, 71–79 (2013)
10. Fukuda, T., Zhang, T., Yabuki, N.: Improvement of registration accuracy of a handheld augmented reality system for urban landscape simulation. Front. Archit. Res. **3**(4), 386–397 (2014)
11. Gaillard, J., Vienne, A., Baume, R., Pedrinis, F., Peytavie, A., Gesquière, G.: Urban data visualisation in a web browser. In: Proceedings of the 20th International Conference on 3D Web Technology, Web3D '15, pp. 81–88. ACM, New York, NY, USA (2015)
12. Guislain, M., Digne, J., Chaine, R., Monnier, G.: Fine scale image registration in large-scale urban LIDAR point sets. Comput. Vis. Image Underst. **157**, 90–102 (2017)
13. Hart, P.E.: How the hough transform was invented [dsp history]. IEEE Signal Process. Mag. **26**(6), 18–22 (2009). https://doi.org/10.1109/MSP.2009.934181
14. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, ISBN: 0521540518 (2004)
15. Johns, D., Dudek, G.: Urban position estimation from one dimensional visual cues. In: The 3rd Canadian Conference on Computer and Robot Vision (CRV'06), pp. 22–22 (2006)
16. Kirchbach, K.: Augmented reality on construction sites using a smartphone-application. In: 2013 17th International Conference on Information Visualisation, pp. 398–403 (2013)
17. Lange, E.: Integration of computerized visual simulation and visual assessment in environmental planning. Landsc. Urban Plan. **30**(1), 99–112 (1994). https://doi.org/10.1016/0169-2046(94)90070-1

18. Luo, J., Etz, S.P.: A physical model-based approach to detecting sky in photographic images. IEEE Trans. Image Process. **11**(3), 201–212 (2002)
19. Meguro, J., Murata, T., Nishimura, H., Amano, Y., Hasizume, T., ichi Takiguchi, J.: Development of positioning technique using omni-directional ir camera and aerial survey data. In: 2007 IEEE/ASME international conference on advanced intelligent mechatronics, pp. 1–6 (2007)
20. Milgram, P.: A taxonomy of mixed reality visual displays. IEICE Trans. Inf. Syst. **E77–D**(12), 1–15 (1994)
21. Nobuyoshi, Y., Shuhei, F., Yuuki, H., Tomohiro, F.: Collaborative visualization of environmental simulation result and sensing data using augmented reality. In: Luo, Y. (ed.) Cooperative Design, Visualization, and Engineering, pp. 227–230. Springer, Berlin (2012)
22. Payam, G., Ian, D.B.: Integration of augmented reality and gis: A new approach to realistic landscape visualisation. Landsc. Urban Plan. **86**(3), 226–232 (2008)
23. Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M.: Geolocalization using skylines from omni-images. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 23–30 (2009)
24. Ramalingam, S., Bouaziz, S., Sturm, P., Brand, M.: Skyline2gps: Localization in urban canyons using omni-skylines. In: 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3816–3823 (2010)
25. Reitmayr, G., Drummond, T.W.: Initialisation for visual tracking in urban environments. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, pp. 161–172 .https://doi.org/10.1109/ISMAR.2007.4538842 (2007)
26. Rune Nielsen Thomas Fabian Delman, T.L.: The harbour game - a mixed reality game for urban planning. In: The Computers in Urban Planning and Urban Management Conference (2005)
27. Sassi, A., Ouarda, W., Ben Amar, C., Miguet, S.: Sky-CNN: a CNN-based learning approach for skyline scene understanding. Int. J. Intell. Syst. Appl. **11**, 14–25 (2019). https://doi.org/10.5815/ijisa.2019.04.02
28. Saurer, O., Baatz, G., Köser, K., Ladický, L., Pollefeys, M.: Image based geo-localization in the alps. Int. J. Comput. Vis. **116**(3), 213–225 (2016). https://doi.org/10.1007/s11263-015-0830-0
29. Schall, G., Wagner, D., Reitmayr, G., Taichmann, E., Wieser, M., Schmalstieg, D., Hofmann-Wellenhof, B.: Global pose estimation using multi-sensor fusion for outdoor augmented reality. In: 2009 8th IEEE International Symposium on Mixed and Augmented Reality, pp. 153–162 (2009)
30. Ventura, J., Hllerer, T.: Wide-area scene mapping for mobile visual tracking. In: 2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp. 3–12 (2012)
31. Yabuki, N., Miyashita, K., Fukuda, T.: An invisible height evaluation system for building height regulation to preserve good landscapes using augmented reality. Autom. Constr. **20**(3), 228–235 (2011). https://doi.org/10.1016/j.autcon.2010.08.003
32. Zhu, S., Morin, L., Pressigout, M., Moreau, G., Servires, M.: Video/GIS registration system based on skyline matching method. In: 2013 IEEE International Conference on Image Processing, pp. 3632–3636 (2013)

Mehdi Ayadi received his engineering degree in Industrial Computing and Automation from National Institute of Applied Science and Technologies, University of Carthage, Tunis, in 2013. He obtained his Ph.D. in Computer Science from the University of Lyon, France, at LIRIS Laboratory. His research interests include augmented reality on mobile in urban context and image processing techniques.

Mihaela Scuturici is an associate professor at Université Lumière Lyon 2, France, and a member of LIRIS Laboratory since 2006. She received her Ph.D. at INSA de Lyon in 2002. She obtained a masters degree in 1996 from the Faculty of Mathematics and Computer Science of "Babes-Bolyai" University, Cluj-Napoca, Romania. Her research interests are image processing and analysis, object detection and recognition in images and videos, object tracking, and motion analysis.

Chokri Ben Amar received B.S. degree in Electrical Engineering from the National Engineering School of Sfax (ENIS) in 1989, and M.S. and Ph.D. degrees in Computer Engineering from the National Institute of Applied Sciences in Lyon, France, in 1990 and 1994, respectively. He spent one year at the University of "Haute Savoie" (France) as a teaching assistant and researcher before joining the Higher School of Sciences and Techniques of Tunis (ESSTT) as an Assistant Professor in 1995. In 1999, he joined the Sfax University (USS) as an Assistant Professor, and since 2011, he is a full professor in the Department of Computer Sciences and Applied Mathematics of the National Engineering School of Sfax. His research interests include computer vision and image and video analysis. These research activities are centered on intelligent algorithms and their applications to data classification and approximation, pattern recognition, watermarking and image and video indexing and securing. He is a senior member of IEEE since 2008. He founded the IEEE Signal Processing Society (SPS) Tunisia Chapter in January 2009, and he is actually the chair of this chapter. During this period, the chapter organized five IEEE Distinguished Lectures and other technical and professional activities. He is the current advisor of the IEEE SPS Student Chapter in ENIS.

Serge Miguet graduated from the École Nationale Supérieure d'Informatique et de Mathématiques appliquées de Grenoble (ENSIMAG) in 1988. He obtained a PhD from the Institut National Polytechnique de Grenoble (INPG) in 1990. He was an Assistant Professor at the École Normale Supérieure de Lyon and a member of the Laboratoire de l'Informatique du Parallélisme (LIP) from 1991 to 1996. He received his Habilitation á Diriger des Recherches from the Université Claude Bernard Lyon 1 in 1995. Since 1996, he is a full Professor in Computer Science at the Université de Lyon, Université Lumière Lyon 2, and a member of the Laboratoire d'Informatique en Images et Systèmes d'Information (LIRIS, UMR CNRS 5205). His main research activities are devoted to models and tools for image processing, image analysis, computer vision, and shape recognition.