Noname manuscript No.
(will be inserted by the editor)

Augmented Songbook: an Augmented Reality Educational Application for Raising Music Awareness

Marçal Rusiñol · Joseph Chazalon · Katerine Diaz-Chito

Received: date / Accepted: date

Abstract This paper presents the development of an Augmented Reality mobile application which aims at sensibilizing young children to abstract concepts of music. Such concepts are, for instance, the musical notation or the concept of rythm. Recent studies in Augmented Reality for education suggest that such technologies have multiple benefits for students, including younger ones. As mobile document image acquisition and processing gains maturity on mobile platforms, we explore how it is possible to build a markerless and real-time application to augment the physical documents with didactical animations and interactive content. Given a standard image processing pipeline, we compare the performance of different local descriptors at two key stages of the process. Results suggest alternatives to the SIFT local descriptors, regarding result quality and computationnal efficiency, both for document model identification and pespective transform estimation. All experiments are performed on an original and public dataset we introduce here.

Keywords Augmented Reality · Document Image Matching · Educational Applications

M. Rusiñol and K. Diaz-Chito
Computer Vision Center, Dept. Ciències de la Computació
Edifici O, Universitat Autònoma de Barcelona
08193 Bellaterra (Barcelona), Spain
E-mail: {marcal,kdiaz}@cvc.uab.es

J. Chazalon
L3i Laboratory, Université de La Rochelle
Avenue Michel Crépeau
17042 La Rochelle cedex 1, France
E-mail: joseph.chazalon@univ-lr.fr

EPITA Research and Development Laboratory (LRDE)
14-16 rue Voltaire
94270 Le Kremlin-Bicêtre, France
E-mail: joseph.chazalon@lrde.epita.fr

1 Introduction

This paper presents the design and the solving of technical challenges related to the development of an Augmented Reality (AR) mobile application which aims at raising awareness and teaching younger children the musical notation, the relation between music, lyrics and animations, and popular songs. This application is called *"Augmented Songbook"*. AR presents very attractive potential benefits for language learning. In our case, the developed application explicits the links between: a document and its embedded message; the musical score, the sound played by some instrument and the keys pressed on a virtual keyboard; the lyrics, the music played, and animations related to the story of the song. Figure 1 illustrates how augmented content is superimposed on a page of the songbook.

We focus in this paper on a key challenge for educational AR tools: the difficulty to maintain superimposed information [3]. Our goal is to present an AR framework suitable for building efficient applications, avoiding poor interaction design which can lead to *"ineffective learning"*, *"disorientation"* and *"cognitive overload"*, according to Yilmaz [37]. This requires to be able, at the same time, to identify which document (among a database of known documents) the user is aiming at, and what is the position of such document on the device's screen. Such process must be tolerant to motion, perspective distortion and illumination variations to face realistic usage conditions. Furthermore, a few extra constraints related to the nature of the project are to be considered: we aim at avoiding marker-based AR technologies to remove all distracting content for the children; and the technologies used must be able to deal with documents containing little text and texture. Such conditions usually hurt significantly the performance of existing image matching techniques.

This paper presents the following contributions, based on the observation that, while document image matching using local descriptors is promising, little work has been published to present how such architecture works and how it performs on real conditions (Section 2):

– we explain how a document image matching architecture can be used to implement an AR platform which can run on mobile devices (Section 3);
– we detail the internals of each key component of this architecture, and present the local descriptor methods under test (Section 4);
– we introduce a new public dataset designed to benchmark our AR platform and enable reproducibility (Section 5);
– we benchmark several local descriptors on realistic data, providing guidance for implementing such system (Section 6).

In order to illustrate how those contributions were integrated in the *Augmented Songbook* project, we also present some of the deployment setups used to enable children and parents to try out this pedagogical prototype (Section 7).

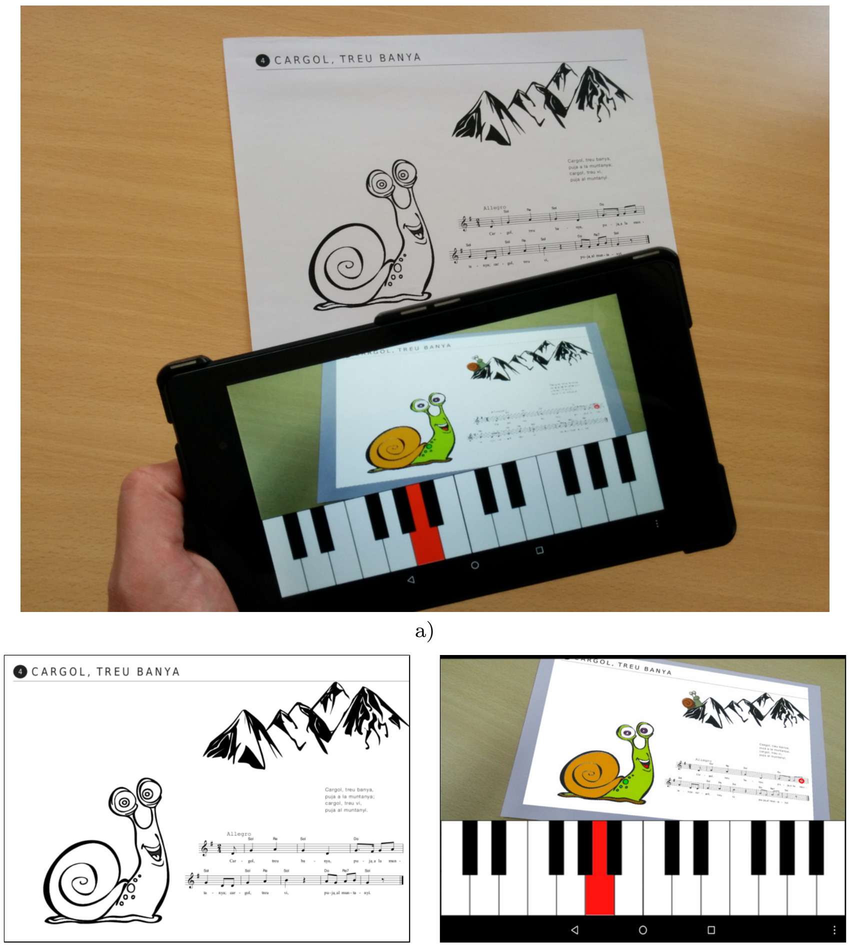


Fig. 1: Illustration of the augmented contents of the proposed application. (a) User's perspective, pointing the mobile device towards one of the documents e.g. (b), enables the augmented contents (c): graphic animations, position of the next note to play and assisted virtual keyboard to play the tune.

2 Background

2.1 Augmented Reality in Education

Recent studies in Augmented Reality for education [3] suggest that such technologies have multiple benefits for students, including younger ones [8]. Both parents [8] and children [37] consistently exhibit a positive attitude regarding well-designed educational applications. Motivation, learning gains, interaction and collaboration are viewed as key advantages [3] of such tools which enable *"superimposing virtual information to real objects"*.

During the last years, a number of the so called technology-enhanced learning applications have been presented for teaching diverse abstract concepts. For instance in [2], AR technology is used to teach geometry concepts, in [5] an AR book is used to teach number to preschool children, in [11] an AR app introduces electronic fundamental concepts, in [22] AR is used to learn about different animals, spatial visualization of calculus concepts is introduced in [29] or the alphabet is presented through an AR app in [30].

Concerning the teaching of musical concepts, a few attempts have been proposed, basically to guide the user to play a certain tune through augmented or completely virtual keyboards, as in [28,14,16,36]. However, to our best knowledge, no AR application has been presented up to now to teach more abstract musical concepts such as musical notation.

2.2 Content-based Document Image Retrieval

In parallel, document image matching techniques have made strong progress over the past years. Those approaches enable a fast identification of a document image (or a part of it) against a database. However, in an AR context we have to take into account that we will face several difficulties specific to camera-based document image processing due to device variability (resolution, sensor quality, optical distortions) and the mobility situation (unpredictable light conditions, perspective distortions, motion, out-of-focus blur), as reported by the organizers of the SmartDoc competition [6,27].

A first notable work is the one of Jain et al. [17] who designed a scalable system for the retrieval of scanned document image which mimics text retrieval, and adds image-specific verifications. First SURF [4] keypoints are extracted from each images, and each descriptor is stored in a hash table to speed-up lookup. Later, when an image is used as a query, SURF keypoints are extracted and matched in descriptor space against closer candidates, in an approximate nearest neighbor fashion. Then, a geometric filtering step is performed to improve precision, first by considering keypoint orientations, then by looking at all combinations of 3 matching points. While limited to scanned documents, and somehow specific to SURF descriptors, this approach exhibits interesting properties as is allows, at the same time, to identify a matching document and to locate the part relevant to the query.

Moraleda and Hull [24,23] followed another approach leveraging a text retrieval engine. Their purpose is to enable content-based image retrieval of textual document from low resolution cellphone images (OCR is not possible here). They associate to each word bounding box a feature vector of variable length which is quantized and stored as a synthetic word in the text retrieval engine. During the retrieval stage, the search engine is used to match close neighbors while tolerating local alterations, and a geometric verification of the relative position of the bounding boxes in retrieved documents is performed to improve precision. While such approach can be partly embedded and can scale to a few thousand images, it is limited to textual documents and required close-up captures with little to no perspective.

Nakai et al. [26] proposed a technique which is very close to fulfill our needs. The authors enable an efficient document image retrieval scheme using camera-captured images as queries. Thanks to an original technique (named LLAH) for detecting and describing local arrangements of keypoints invariant to perspective distortion, a simple voting system enables the retrieval of documents given a camera-captured excerpt of the original document. The improvements proposed by Takeda et al. [33] unlocked the scaling of their method to millions of images as well as locating precisely the part of the original document images contained in the query (using the RANSAC [13] to find a geometric consensus) for estimating the perspective transform. The authors even experimented some AR techniques [34] on a smartphone to superimpose information on the document aimed at. However, this approach has two main drawbacks which prevent using it for our project which aims at retrieving graphical documents without any external dependency: first, the local descriptor used is specific to text documents, and cannot cope with documents which are mostly graphical and contain very little text; and second, the system architecture used relies on a network connexion to query a distant computer in real time, in order to cope with RAM limitations on a mobile device.

Due to their flexible matching properties, approaches based on local descriptors are the most promising to face the challenges of the *Augmented Songbook* project. In particular, such techniques can cope with partial occlusions, degraded document, and difficult capture conditions: motion and out-of-focus blur, perspective, too much or too little light, etc.

In this paper, we are particularly interested in identifying a suitable architecture which enables the simultaneous identification and location of the document under capture on a mobile device, and evaluating the performance of such system under realistic conditions.

3 Mobile Document AR Using Local Descriptors

This section introduces the architecture of the mobile document AR application. It is simple and proposes a clear definition of its processing steps, hence enabling a precise evaluation of each of them.

The architecture used in the *Augmented Songbook* project forms a real-time image processing pipeline. On the mobile device, the camera subsystem generates a stream of video frames which must be processed as quickly as possible in order to render smoothly the appropriate augmented content over the scene under visualization.

Figure 2 provides a general view of the information flow between the main components of the application:

- the **Camera Module** streams frames to all the other modules in real time;

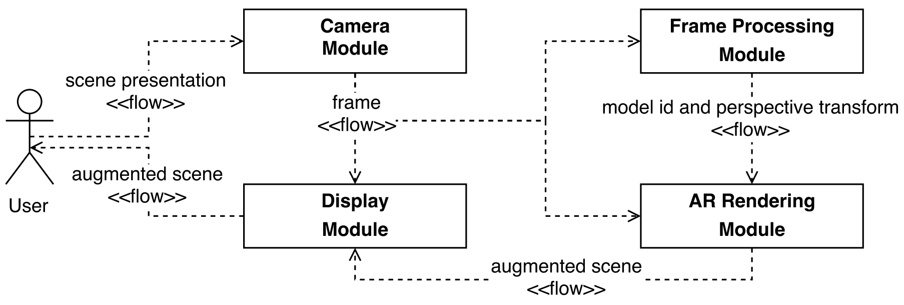


Fig. 2: Global information flow among the components of the application. Each module runs independently of the others.

- the **Frame Processing Module** checks for any known document in the scene, and eventually locate its position;
- the **AR Rendering Module** uses information about document model and position to render the animation with the appropriate timing and projection;
- the **Display Module** finally blends all the content (animation, widget, original frame) in real time.

Each module runs independently of the others, enabling the Camera Module and the Display Module to work at full speed at OS level (on mobile platform), while allowing the Frame Processing Module and the AR Rendering Module to process content at their own pace.

In this paper, we are interested in the Frame Processing Module, though we will briefly describe the AR Rendering Module, which is very rudimentary in our case.

3.1 Data Flow at Run Time

Data processing and flow within the AR architecture is illustrated in Figure 2. It presents the main actions and objects at an abstract level. It also shows that the workload can be distributed across several independent threads. As mentioned previously, such distribution permits to provide the user with a real-time display even if Frame Processing and AR Rendering are too slow to process every frame acquired by the camera. We detail here the building blocks of the architecture, and will describe how they collaborate in real time in section 3.2.

At the **operating system level**, the following actions are performed.

- **Read Frames** to produce a continuous stream of images and their timestamps.
- **Display the AR Scene** to the user, blending animated content, original scene and various widgets.

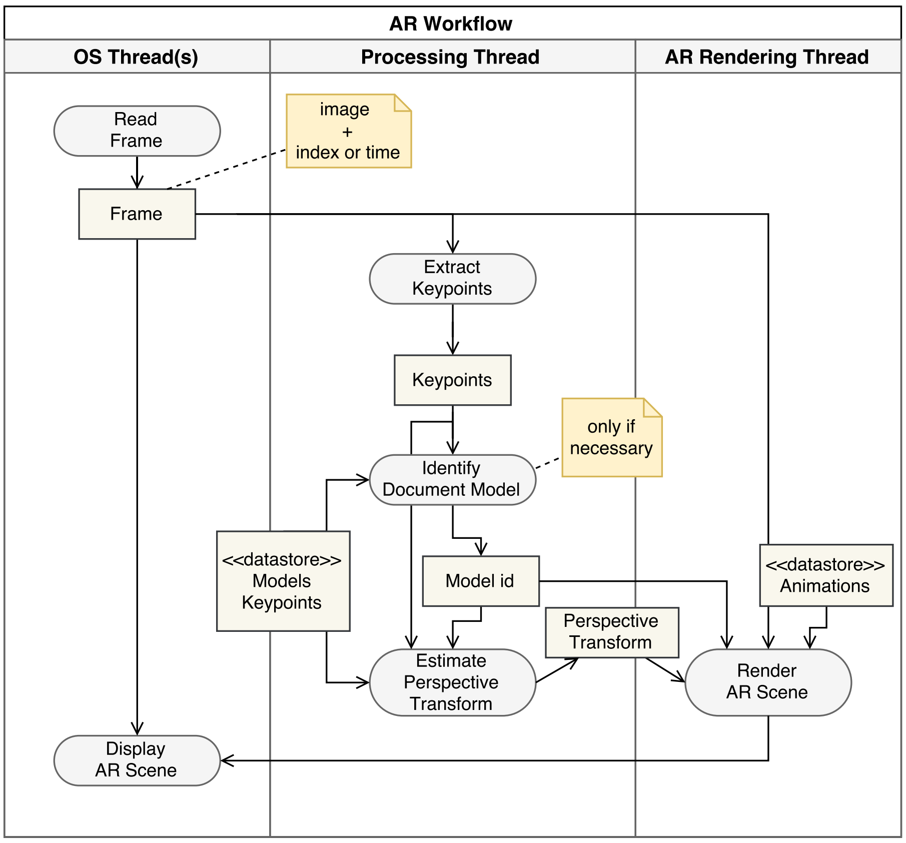


Fig. 3: Data flow at run time. Control flow is removed for clarity. Actions are represented by rounded rectangles, and objects by squared ones. This illustration completes the one in Figure 2 by detailing the content of the Frame Processing Module and showing explicitly the distribution of actions across threads.

The core of the architecture is the **frame processing** comprised of the following actions run sequentially.

– **Extract Keypoints** from frame images. This consists both in detecting salient and robust points or regions, and computing a descriptor to characterize the related local area of the image. Such descriptors must be as robust as possible to the common distortions encountered during mobile document capture: perspective distortion, illumination variations, out-of-focus blur and motion blur, in particular. The **Keypoints** object is therefore a set of structures containing coordinates, a description vector, as well as a few extra elements related to the implementation.
– We then **Identify the Document Model** using a) the set of keypoints extracted from the current frame; and b) the set of all keypoints extracted

from every document model image (named here **"Models Keypoints"**, which stores for each keypoint the id of the associated document model). The decision process leading to the selection of a single document model will be detailed in Section 4.2.
– Finally, we **Estimate the Perspective Transform** making use of the correspondences between: a) the set of keypoints extracted from the current frame; and b) the subset of **Models Keypoints** which correspond to the model which was identified. It consists in recovering the transformation matrix describing the perspective under which the document model is viewed. It also serves as a geometric validation stage of the previously matched document model, rejecting inconsistent solutions.

The identification of the document model is performed only when no document model has been found yet, or when the perspective transform estimation fails for a given number of frames, indicating that the document in the scene may have changed. Conversely, there is no need for a specific model identification as long as the perspective model estimation succeeds (indicating that the document model is still valid for the current frame).

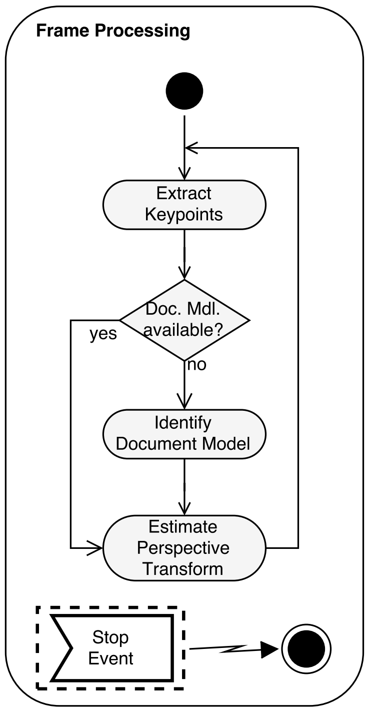
AR Rendering, finally, can be reduced to a single action for the purpose of our presentation: **Render the AR Scene**. This action consists in superimposing the appropriate animation for the current model, using the perspective transform previously found, on the current frame. It uses a bank of animations (**Animations**) which contains images, sounds, and other elements for each document model. The main challenge of this step is to maintain a smooth animation display with a high frame rate while:

1. frame processing introduces a sensible delay between the moment when a frame is read, and the moment when the perspective transform is available;
2. occasional failures in document model identification or perspective transform estimation create gaps in the availability of the perspective transform.

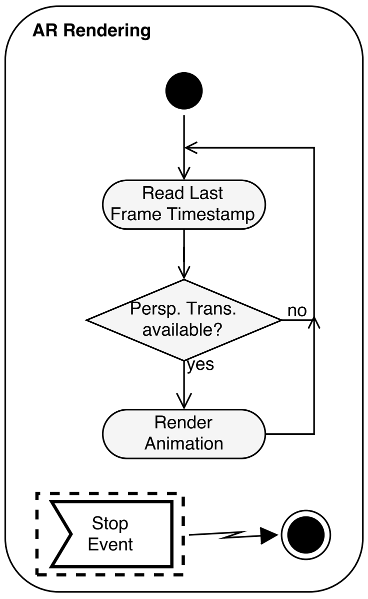
To cope with those issues, the rendering has to keep track of the last valid model id and perspective transform in order to avoid flickering effect (intermittent display of augmented content). Furthermore, it is possible to improve the quality of the rendering by interpolating the perspective transform using the frame timestamps and a fast motion estimation technique, like optical flow [21] for instance. As such technique introduces an extra computing cost, it can have a negative impact on some configuration; and would therefore deserve a dedicated study.

3.2 Control Flow at Run Time

As frames have to be processed as quickly as possible, the frame processing and the AR rendering perform a minimal amount of work at each cycle, as illustrated in figure 4.



a)



b)

Fig. 4: Activity diagrams of a) frame processing and b) AR rendering.

For **Frame Processing**, keypoint extraction, document model identification and perspective transform estimation are performed sequentially. However, document model identification is performed only when no document model is already available. The reference to the document model is kept as long as the geometrical validation of the perspective transform estimation succeeds. If the perspective transform estimation fails for a sufficient number of times, then the document model is invalidated as it may indicate that the user is now aiming at a different document, and document model identification will be run on the next iteration.

AR Rendering, on the other hand, only renders the animation when a perspective transform is available. Like for document model identification, the last valid perspective transform is stored and applied to later frames until a new transform is found, or until the estimation failed for a sufficient number of times.

The clear distinction between document model identification and perspective transform estimation permits to reduce significantly the amount of time required to process a frame as soon as a model is found and validated. Keypoint extraction is however a required operation for each new frame to process. Figure 5 illustrates the benefit of such separation on a sample session: as soon as the document model is found and validated, perspective transforms are computed much faster, enabling a smoother AR rendering.

4 Implementation of the Frame Processing Module

We now focus on the three main processes involved in the core of the architecture; the frame processing. Keypoint extraction (section 4.1) is the fundamental process which consists in producing a set of local descriptors for each video frame. This set is latter matched against either i) the full database of local descriptors ("Models Keypoints") previously extracted from all document models (during Document Model Identification, Section 4.2) or ii) a subset of this database of local descriptors (during Perspective Transform Estimation, Section 4.3).

This global process is very robust against occlusions or partial detections, due to missing or invisible parts of the document, because of the nature of the set comparison techniques used to match the set of keypoints extracted from each frame against the set of keypoints previously extracted from model images.

4.1 Keypoints Extraction

The keypoint extraction process is based on two distinct stages:

1. a detection stage which identifies interest points or regions which exhibit saliency properties and should be stable under multiple viewing conditions: perspective, illumination, motion, and focus in particular;
2. a description stage which encodes visual information (texture, gradients, colors, etc.) about the neighborhood of the interest element with a transform function which is invariant to several of the previously mentioned distortions.

We considered four methods for keypoint detection and description for our analysis.

- **SIFT**: Keypoint detector proposed by D. Lowe in [20] in which keypoints are extracted as maxima of the Difference of Gaussians over a scale space analysis at different octaves of the image. Dominant orientations are assigned to localized keypoints. The descriptor coarsely describes edges appearing in keypoint frames by an orientation histogram over the gradient image.
- **SURF**: Keypoint detector proposed by H. Bay et al. in [4] which detects blobs based on the determinant of the Hessian matrix. The descriptor is based on the computation of Haar wavelet responses in a dense fashion within the keypoint frames.
- **ORB**: Keypoint detector proposed by E. Rublee et al. in [31] which uses an orientation-based implementation of the FAST corner detection algorithm.

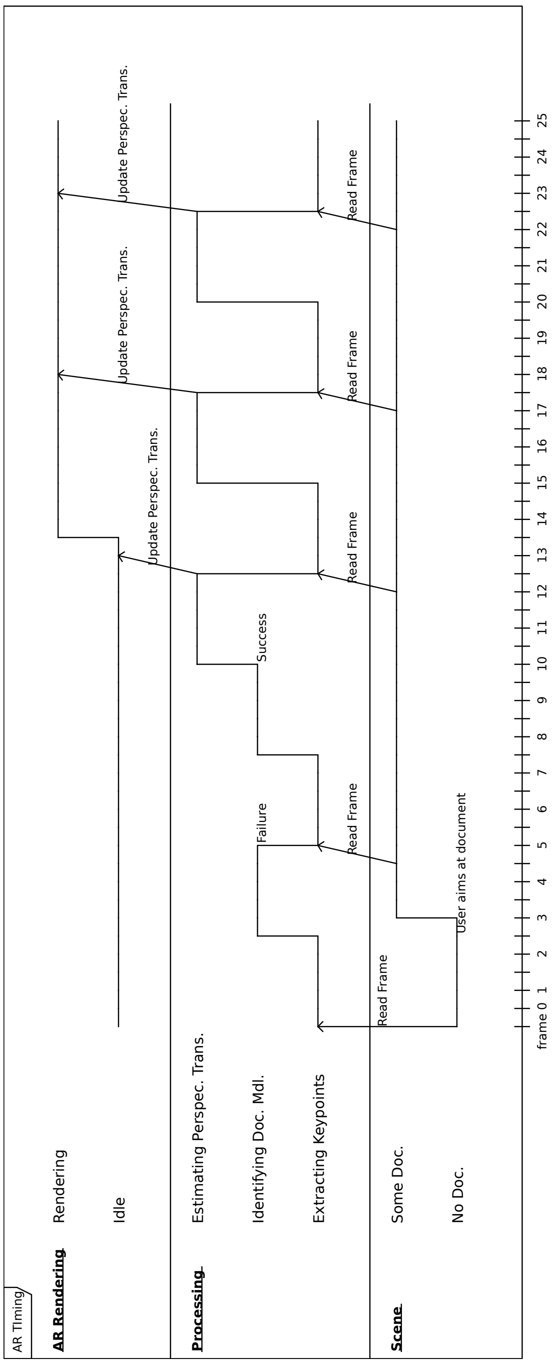


Fig. 5: Timing diagram of a sample AR session. At the beginning, no document is visible in the frame and document model identification fails. Starting from the fourth frame, a document in visible in each frame. This enables the document model identification, and subsequently the perspective transform estimation, to succeed. Once document model identification is confirmed by the geometrical validation of the perspective transform estimation, there is no need to check for a new document model unless perspective transform estimation fails for a sufficient number times. As soon as a valid perspective transform is available, the AR rendering becomes active.

The binary descriptor is a rotation aware version of the BRIEF descriptor [7]. It basically encodes the intensity differences among a set of pairs of pixels.

– **BRISK**: Keypoint detector by proposed by S. Leutenegger et al. in [19] which analyses the scale space using a saliency criterion. The binary descriptor is also based on a pair-wise comparison of pixel intensities as in ORB.

It is worth noting that both SIFT and SURF yield an integer-valued histogram while ORB and BRISK produce binary strings. Such binary descriptors are matched against each other with a Hamming distance which entails a much faster distance computation than the Euclidean distance calculation done for SIFT and SURF descriptors. ORB and BRISK being binary descriptors they are more suitable for a real-time applications in mobile devices since they offer matching speeds that can yield several orders of magnitude in speed when compared with integer-valued descriptors. In this paper we analyze their performances in order to assess the performance loss in terms of accuracy when using such compact descriptors.

4.2 Document Model Identification

We followed a standard architecture for document matching with local descriptors [9,32]. Given a set of model documents $D = \{d_1, d_2, ..., d_M\}$ to index, we compute local detectors to end up with N keypoints $K = \{k_1, k_2, ..., k_N\}$ for each model document from D. Given the different detectors we tested, the keypoints from K are to some extent invariant to rotation, scale, illumination and perspective changes. Each keypoint k_i is then described by a feature descriptor f_i from one of the previously presented descriptors. Such descriptors are then indexed in an inverted file efficiently implemented with the FLANN architecture [25], which either uses KD-trees for integer descriptors (like SIFT and SURF) or LSH [1] for binary descriptors (like BRISK and ORB). Each entry of the model database is composed of keypoint information: coordinates, local descriptor value and original model identifier.

For any incoming frame from the mobile device camera, keypoints and local descriptors are extracted and computed, and matched against the inverted file. In order to produce reliable matches, we use the ratio-test proposed by Lowe [20], according to which a match is considered to be correct if the ratio between the distance to the nearest and to the second nearest local descriptors is above a certain threshold. An extra level of control would be possible by considering reverse matches between descriptors from either the complete database or only the selected document model and descriptors from the current frame, but our experience showed that this tends to filter too many candidates, sometimes even degrading the final decision while adding a significant computational cost. The model selected as appearing on the frame is the one for which more local descriptors have been matched without ambiguity. For efficiency reasons, this architecture does not allow for more than one document model to be matched.

4.3 Perspective Transform Estimation

Perspective transform estimation is performed during a second keypoint matching stage. Once the document model is known, it is possible to reconsider all the local descriptors extracted from the frame and match them against the ones from the same model in the model database. A ratio test is also performed at this stage to filter ambiguous matches. From a set of putative matches between keypoints of the selected document model and of the current frame, a RANSAC [13] step is performed in order to filter out the outlier matches that to not agree geometrically and to find the homography between the recognized model document and its instance appearing in the scene. We found that a minimum of 15 inliers was required to obtain stable results.

This stage confirms the validity of the document model which was identified at the previous stage, and provides the AR rendering module with a transformation matrix which allows to project the augmented synthetic content over the physical document in the current frame.

We separate this stage from document model identification as much as possible, in the sense where we take all keypoints from the matched document model and match them against all the keypoints of the frame. This solution is slower (as it requires a second approximate nearest neighbor search, even if restricted to a subset of the model database) but provides better results when compared to solutions which estimate the perspective transform directly on keypoints correspondences obtained from the document model identification stage. As previously mentioned, this separation is the key to a clear evaluation of this architecture.

5 Public Dataset for Performance Evaluation

This section first describes the content of the test database we created for this project[1], and then details the ground-truth creation process.

5.1 Database contents

In order to create our dataset, we built fifteen children musical sheet pages taking as a basis the score sheets published in an official monograph from the education department of the Goverment of Catalonia [2]. Our sheets contain the title of the song, the music score, the lyrics of the song as well as some

[1] `http://www.cvc.uab.cat/songbook/`

[2] `http://ensenyament.gencat.cat/ca/departament/publicacions/monografies/cancons-populars`

simple drawings depicting the main content of the song. We show an example of each page in Figure 6.

Each of those document models were printed using a laser-jet printer and we proceeded to capture them using a Google Nexus 5 smartphone. We recorded small video clips of around 20 seconds for each of the fifteen documents in three different acquisition conditions presenting severe perspective distortion, low contrast between background and musical sheet, motion and out-of-focus blur, etc. The videos were recorded using Full HD 1920x1080 resolution at variable frame-rate. Since we captured the videos by hand-holding and moving the smartphone, the video frames present realistic distortions such as focus and motion blur, perspective and change of illumination. In addition of the video clips, we also captured an 8Mp picture of each of the documents to be used as models for the matching process. We present an example of the different scenarios in Figure 7. Summarizing, the database consists of 45 video clips (of 18.7 seconds on average) comprising 21 048 frames.

For each frame of the dataset, one and only one instance of a document is fully visible: no corner or side is “cut”, and the coordinates of each of the 4 corners of the page in the frame referential are stored in the ground truth.

5.2 Semi-automatic groundtruthing

To evaluate the ability of the proposed approach to locate and accurately segment the document pages in video feeds, we need a segmentation groundtruth of our collection consisting in having a quadrilateral defined by the four corners of the paper sheets appearing in the video frames. However, manually annotating such amount of frames, is a tedious and error-prone task. We have used the semi-automatic ground-truthing approach previously presented in [10]. We start with the assumption that the documents lie on a flat surface and do not move from it. We surround the physical document with four color markers that will be easily segmented. The ground-truth quadrilateral is inferred through computing the transformation that holds between the markers and the paper sheet. Such approach involves a reduced human intervention.

The proposed semi-automatic approach works as follows. First color markers shown in Figure 8a) have to be segmented. Given a video with n frames $F_0, ..., F_i, ..., F_{n-1}$, we show the first frame F_0 to the user that has to click on the four markers. The RGB color values in those positions with a range tolerance are used as thresholds to segment the markers, as shown in Figure 8b). Such RBG values are update iteratively for each frame to tackle illumination changes across the whole video, i.e. the RGB value of each of the markers centroid at the ith frame is used to segment the $i + 1$th frame.

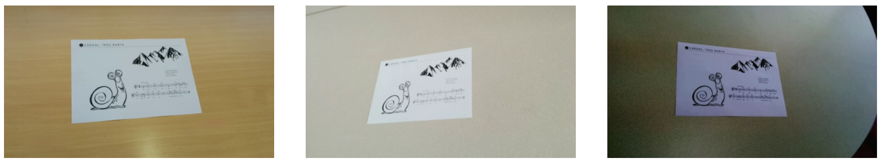
Being M_i the polygon defined by the marker centroids $M_i = \{A_i, B_i, C_i, D_i\}$ and P_i the quadrilateral defined by the four page corners $P_i = \{W_i, X_i, Y_i, Z_i\}$ for the ith frame (Figure 9), we define a reference coordinate system by four points $M' = \{A', B', C', D'\}$ in order to compute for each frame a perspective

Fig. 6: Sample documents used in our dataset.

transform [15] H_i that converts the point set M_i to M' using

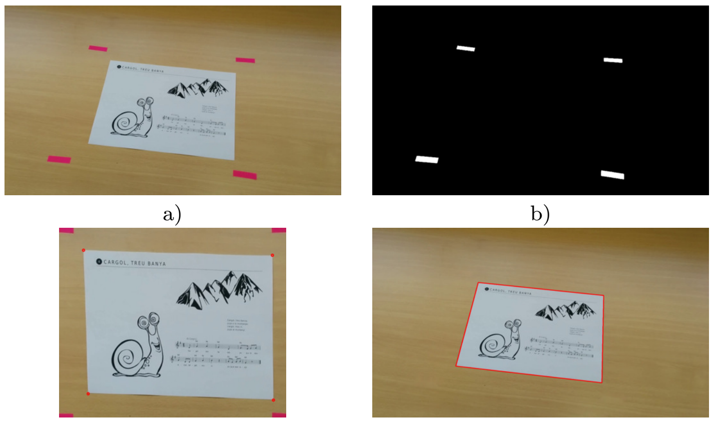
$$M' = H_i M_i.$$

H_0 is computed for the first frame of the video and we present the warped image F'_0 to the user. He then selects the four corners $P' = \{W', X', Y', Z'\}$



Scenario 01 Scenario 02 Scenario 03

Fig. 7: Example of the different acquisition scenarios.



a) b) c) d)

Fig. 8: Overview of the semi-automatic groundtruthing approach. a) Original image. b) Color marker segmentation. c) Warped frame shown to the user to mark the four document corners. d) Quadrilateral output and inpainted markers.

(Figure 8c). Backwards projecting the points from P' using the inverse perspective transform H_i^{-1},

$$P_i = H_i^{-1} P',$$

is used to find the corners of the page P_i at any frame i.

With this approach, we annotated the whole dataset by needing eight clicks from the user per video.

Finally, the marker segmentation mask is used in order to “erase” the markers from each frame using an inpainting technique. This step is optional and is just used to provide markerless and aesthetical video frames. We have used the approach by Telea [35]. We can see the page segmentation and the marker inpainting results in Figure 8d).

A final manual revision of the ground-truth polygons was conducted. Inspecting the ground-truth we found that for a few clips for which color marker segmentation failed, a manual correction was necessary and introduced an extra edition cost. This was detected thanks to a frame by frame inspection of the final video clips (after inpainting). The averaged total time needed to gen-

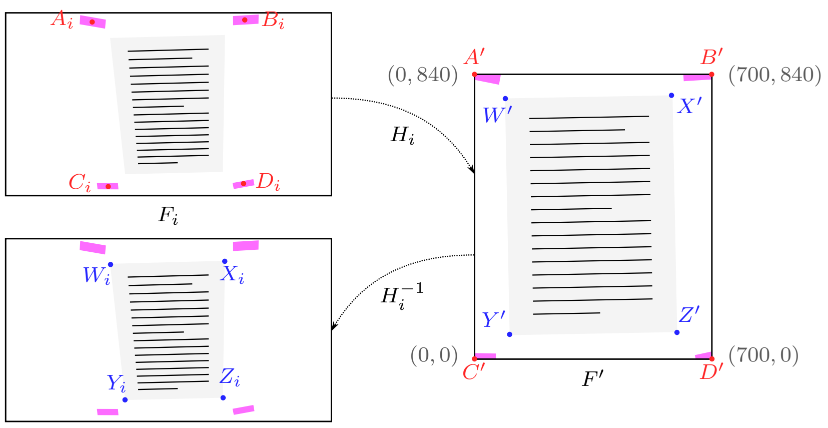


Fig. 9: Markers-to-corners approach. Marker centroids $\{A_i, B_i, C_i, D_i\}$ are detected and H_i mapping those points to know coordinates $\{A', B', C', D'\}$ is computed. The page corners $\{W', X', Y', Z'\}$ within this referential F' are stable. The inverse transformation using H_i^{-1} gives the real coordinates $\{W_i, X_i, Y_i, Z_i\}$ of the corners. (Image reprinted from [10]).

erate the ground truth for a given video clip and its inpainted counterpart is 6 minutes. This includes the capture, the automated processing time, and the manual inspection and correction of all frames to ensure a precise result. This makes our approach very competitive to generate realistic datasets with little constraints.

6 Benchmark of Local Detectors and Descriptors

In this section we will evaluate the performance of the different local descriptors both for the document model identification and for the perspective transform estimation tasks.

We will first start this section by providing some practical details about each local descriptor method used in the benchmark. Then, for each task we will present the evaluation metrics and evaluation protocol to end up presenting and discussing the results.

6.1 Local Descriptors Analysis

The results presented in the rest of this section should be interpreted while keeping in mind the actual amount of local descriptor used to perform each task, and the real memory impact of each method. Table 1 summarizes the memory occupation required to store a single descriptor with each of the techniques we studied. Table 2 indicates the average amount of keypoints extracted

Table 1: Comparison of memory occupation for storing a descriptor.

Method	Bits per dimension	Number of dimensions	Total descriptor size (bytes)
BRISK	8	64	64
ORB	8	32	32
SIFT	32	128	512
SURF	32	64	256

Table 2: Comparison of actual number of descriptors used and associated memory occupation at runtime.

	Keypoints count		Memory usage (in KiB)		
Method	per model	per frame	for all 15 models	per frame	total
BRISK	369	492	346	31	377
ORB	1,020	1,010	478	32	510
SIFT	935	1,570	7,013	785	7,798
SURF	522	855	1,958	214	2,171

either from each model, or from each video frame to be processed. It also provides a rough estimation (not including indexation structure) of the average memory occupation required to store model and frame descriptors at run time. It is worth mentioning here that the model image size is slightly smaller than the one of the video frames to improve the matching (because actual documents rarely cover the whole video frames in the test set).

As we can see, ORB and BRISK exhibit much lower memory impact at run time. Another interesting point to mention is that there are important variations in the number of keypoints extracted by each methods, ORB and SIFT producing more elements to match. This obviously has an important impact on both the quality of the results and the actual processing speed. Regarding ORB and BRISK, the two methods which appear as the most suitable for an embedded solution, we can ask ourselves whether ORB (which produced a target number of keypoints) supports are representative enough for matching frame and model content, and BRISK lacks some of them, or on the contrary if ORB introduces noise and BRISK is well-focused on the essential supports. As we will see in the following of this paper, the results are clearly in favor of using ORB in favor of BRISK for the task we defined.

6.2 Document model identification

The document model identification task is defined as the process of associating to a given frame from the test set the appropriate document model identifier, given a database of keypoints for all document models.

We considered each frame as an independent event, in the sense where we did not implement a solution which made use of the label associated to previous frames in a stream, in order to compare the impact of the choice of local detectors and descriptors, and not regularization techniques (which should be studied separately). However the evaluation model is compatible with more elaborate modules which would take past history into account.

6.2.1 Metrics

Formally, each process will be evaluated according to 2 metrics:

1. on its accuracy on the identification task consisting in association to a random frame the appropriate document model identifier among the fifteen possible ones in the dataset;
2. on its average frame processing time.

The identification accuracy is defined as the number of frames correctly labeled, normalized by the total number of frames.

6.2.2 Protocol

This experiment compares the metrics obtained for each pair of local detectors and descriptors presented in Section 4.1. The experiments were run on a desktop machine (as opposed to mobile devices) for practical reasons. This has no impact on the accuracy metric, but the average frame processing time can be reduced, therefore this last metric should be interpreted relatively.

6.2.3 Results and discussion

We report in Table 3 the obtained identification accuracies and the required processing times for all the local keypoint detectors and descriptors. Even if SIFT outperforms the rest of local descriptors in terms of identification accuracy, it is worth noting that it is the most computing demanding one. ORB achieves comparable identification results while being much faster. We experience a slight drop in performance when using SURF, while BRISK performs poorly on this task despite being the fastest method. The speed difference between using BRISK or ORB is explained by the different amount of keypoints that are extracted from each frame and each model, that we presented in Table 2. From these results we conclude that for a real-time application ORB would be the preferred choice for its speed and identification accuracy.

We report in Tables 4 and 5 the averaged identification accuracies for the different acquisition scenarios and document types respectively. On the one hand we can see that for both SIFT and ORB, the second scenario is the one that causes more failures, since it is the scenario with the most difficult acquisition conditions (perspective, blur and low contrast). On the other hand, by looking at the different document types, we can see the strong variance for BRISK: for some documents (e.g. "10 *sardana*", "12 *gallina*" or "15 *bondia*") it

Table 3: Performance summary on the identification (document model identification) task.

Method	Accuracy (%)	Average FPS	Average time per frame (s)
BRISK	87.3	19.6	0.051
ORB	99.5	4.1	0.241
SIFT	99.9	0.4	2.669
SURF	94.7	0.9	1.083

Table 4: Document identification accuracy per acquisition scenarios.

Scenario	BRISK	ORB	SIFT	SURF
01	79.9	99.9	100.0	92.2
02	90.5	99.0	99.7	98.8
03	91.5	99.6	99.9	93.2
All	87.3	99.5	99.9	94.7

Table 5: Document identification accuracy per document.

Document	BRISK	ORB	SIFT	SURF
01 sol	98.8	99.5	100.0	99.9
02 lluna	57.6	100.0	99.8	72.8
03 plou	84.9	99.2	99.6	99.6
04 cargol	93.4	99.6	99.9	100.0
05 pedra	59.7	100.0	100.0	100.0
06 gegant	87.9	99.1	100.0	100.0
07 jan	100.0	100.0	100.0	100.0
08 olles	90.4	98.2	100.0	98.3
09 quinze	69.9	100.0	99.6	50.3
10 sardana	99.2	99.5	100.0	100.0
11 tres	99.1	100.0	100.0	100.0
12 gallina	99.1	99.3	100.0	100.0
13 cotxe	76.9	99.8	99.9	100.0
14 carrer	93.8	99.4	99.5	99.7
15 bondia	99.3	99.4	100.0	100.0
All	87.3	99.5	99.9	94.7

performs quite similarly to ORB, while for some others its performance drops dramatically (e.g. "02 *lluna*"). The same applies between SIFT and SURF, which perform poorly on a couple of documents. Again, there are no significant differences between SIFT and ORB, but SIFT steadily delivers slightly better accuracies.

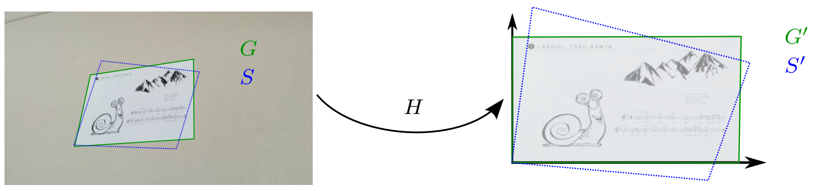


Fig. 10: To ensure that expected (G) and result (S) surfaces are comparable among frames, surface comparisons are performed within the referential of the document, leading to new quadrilaterals G' and S' after dewarping the coordinated using the homography H associated to the current frame F, as described in Figure 9.

6.3 Perspective transform estimation

The perspective transform estimation task consists in finding the best possible perspective transform, given a document model image and a video frame, which will project the synthetic augmented content on the physical document with as little overlapping error as possible.

Here we start from the already known document model, in order to avoid any bias from this prior step.

6.3.1 Metrics

As for document model identification, we consider two metrics:

1. perspective estimation quality;
2. average frame processing time.

To evaluate perspective estimation quality, we used the Jaccard index measure [12] that summarizes the ability of the different methods at correctly segmenting page outlines while also incorporating penalties for methods that do not detect the presence of a document object in some frames. Here we add a preliminary step to project ("dewarp") point coordinates into the document referential, so as to be able to compare values obtained for different frames.

The evaluation procedure works as follows. Using the document size and its coordinates in each frame as stored in the ground truth, we start by transforming the coordinates of the result quadrilateral S and of the ground-truth G to undo the perspective transform and obtain the corrected quadrilaterals S' and G', as illustrated in Figure 10. Such transform makes all the evaluation measures comparable within the document referential, or, said differently, it projects the expected and detected frame regions into a space where each pixel accounts for the same physical surface. For each frame F, we compute the Jaccard index (JI) that measures the goodness of overlapping of the corrected quadrilaterals as follows:

$$JI(F) = \frac{area(G' \cap S')}{area(G' \cup S')}$$

Table 6: Performance summary on the perspective transform estimation task.

Method	Mean Jaccard Index (%)	Average FPS	Average time per frame (s)
BRISK	77.7 ± 0.54	13.5	0.074
ORB	98.4 ± 0.09	9.6	0.104
SIFT	98.8 ± 0.06	0.6	1.741
SURF	97.5 ± 0.11	1.1	0.886

where $G' \cap S'$ defines the polygon resulting as the intersection of the detected and ground-truth document quadrilaterals and $G' \cup S'$ the polygon of their union. As the dataset contains only clips where every frame represents one (and only one) of the documents, the case where $G' \cup S' = \emptyset$ never occurs, thus ensuring the metric is always defined in the experiments we conducted.

Finally, the overall score for each method will be the average of the frame score, for all the frames in the test dataset.

6.3.2 Protocol

This experiment compares the metrics obtained for each local descriptor presented in Section 4.1: BRISK, ORB, SIFT and SURF. As for the previous experiment, this experiment was run on a desktop machine so the average frame processing time should be interpreted relatively.

6.3.3 Results and discussion

We report in Table 6 the quality of perspective transform estimation and the associated average frame processing time for each local descriptor. While SIFT outperforms other methods from a quality perspective, ORB achieves comparable results with a fraction of the computing power required: processing a frame with ORB is more than 16 times faster, while using 16 times less memory to store the same amount of descriptors. SURF performs just a little less well that ORB and SIFT, but at the price of an important processing time, making it less attractive than ORB. BRISK, finally, is very fast, but leads to poor results and is unusable for this task.

During our tests, we realized that methods for which the Jaccard index drops below 95% do not perform well enough for the document augmentation task, as the detected region misses too much information from the document and has a visible overlapping difference for the user. This lets us with only two viable options, SIFT and ORB, out of the four methods we considered. SIFT should be preferred when computational power is not a concern and quality is of prime interest. ORB should be preferred for mobile applications when real-time image processing is performed on the device. Figure 11 provides a more detailed view of the actual distribution of the values of the metric.

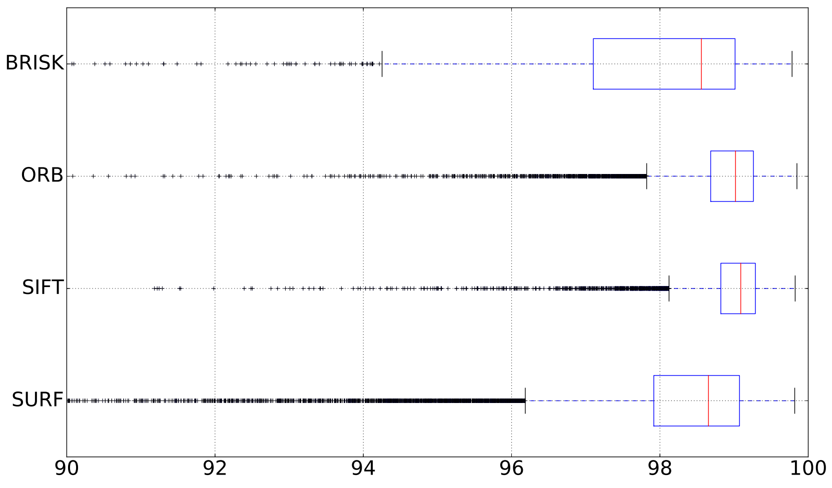


Fig. 11: Boxplot visualization of the distribution of Jaccard index values (in percentages) for each method over all test frames, truncated at 90%.

Table 7: Quality of the perspective transform estimation detailed by acquisition scenario.

Scenario	BRISK	ORB	SIFT	SURF
01	71.9	98.9	99.1	98.3
02	79.8	97.7	98.3	96.4
03	81.4	98.6	98.9	97.9
All	77.7	98.4	98.8	97.5

Finally, the comparison of the performance of each method against acquisition scenarios and documents of the dataset can reveal some additional information. Table 7 details the results for perspective transform estimation for each acquisition scenarios. The second scenario tends to contain more blurry frames, due to a pale background and a lower ambient light which are more challenging for the autofocus. It also contains frames with more perspective distortion. With the exception of BRISK, the methods seem to be more impacted by such factors. It is interesting to note that the overall ranking remains the same when capture conditions change: SIFT performs slightly better than ORB, SURF then performs reasonably before BRISK for which performance is too low. Table 8 details the results for each document of the dataset. Some documents, like "09 *quinze*" contain little texture information and are challenging for all methods. Depending on documents, and maybe also on acquisition samples, the order between SIFT and ORB can be switched, and also sometimes between ORB and SURF. SIFT however always performs better than SURF. BRISK, on its side, constantly performs worse despite its great variance: it

Table 8: Quality of the perspective transform estimation detailed by document.

Document	BRISK	ORB	SIFT	SURF
01 sol	83.8	98.1	98.6	97.8
02 lluna	60.8	98.7	98.4	96.3
03 plou	75.3	97.7	98.4	96.5
04 cargol	88.5	98.5	98.8	98.6
05 pedra	41.7	98.9	98.6	98.0
06 gegant	86.3	98.4	99.2	98.1
07 jan	92.4	98.8	98.9	98.4
08 olles	82.4	97.0	98.8	94.3
09 quinze	20.8	98.6	98.5	95.9
10 sardana	97.0	98.8	99.1	98.7
11 tres	86.6	99.0	99.2	98.3
12 gallina	91.7	98.1	98.8	98.3
13 cotxe	75.6	98.9	99.2	98.4
14 carrer	87.9	97.7	98.0	96.4
15 bondia	96.2	98.8	99.2	98.8
All	77.7	98.4	98.8	97.5

can provide decent results for some documents like "10 *sardana*", with 97.0% average JI, and unusable ones for documents likes "09 *quinze*", with 20.8% average JI.

To conclude, SIFT is a clear winner, but ORB is the method of choice for mobile applications, with a small quality loss. However, under more challenging conditions like low light, motion and defocus blur, or occlusions, the game may change and this ranking could be more severe. In our experience, we observed that SIFT can handle difficult cases where ORB performance drops, before giving up on its turn.

7 Application Details and Implementations

We built a prototype of the application in Java using the OpenCV's wrapper for Android. The app automatically detects the music score we are pointing at from a reduced dataset of three indexed scores. At this stage, the black and white drawings from the music score documents are augmented with a colorized version. The user can select either to use a piano, a violin, a flute or a saxophone as playing instrument and a virtual keyboard appears. The user is then guided to play the song by marking both the key to play from the keyboard and the corresponding note from the staff. Once the correct key has been played, the next key and note is displayed. This aims thus at acquiring abstract musical concepts by a manipulative and interactive application. We can see an example of our prototype in Figures 1 and 12. A couple of videos of the usage of the app can be seen in the website of this project.

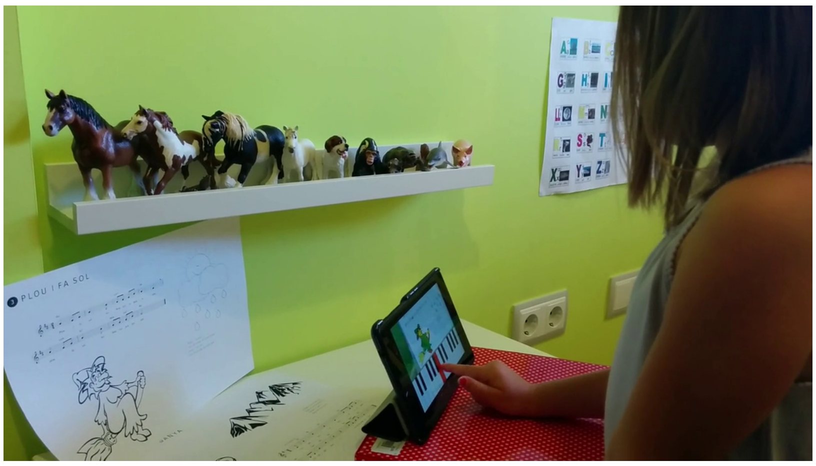


Fig. 12: An example of usage of the developed AR prototype.

The processing thread in the current non-optimized version[3] of the prototype runs at 7 *fps* in a Google Nexus 7 tablet. Such latency might not be enough for a pleasant use, since the users will experiment a lag. However, by having two separate threads, one for processing, and one for the camera grabbing and display modules, as explained in Section 3, provides the user an enjoyable experience with a 23 *fps* feedback sensation.

It would be very interesing to really assess and evaluate how such new learning applications are perceived by the users and if they actually contribute in raising awareness of the musical notation to young children, and to ease the learning of abstract concepts. Although we have tested its use with several kids aged around 6 years old, receiving positive reactions, we have not yet conducted such in-depth evaluation.

Finally, we have a permanent installation of the system in the Volpalleres Library Living Lab in the town of Sant Cugat, Spain, for the general public to test it [18].

8 Conclusions

In this paper we described the architecture and the internals of a mobile educative application devoted to raise awareness of the musical notation to young children. The application allows the superimposition of augmented contents over pages of a predefined songbook in real time, without specific markers, and with all the processing done on the mobile device.

We have benchmarked the performance of local detector and descriptor methodologies for the tasks of document model identification and perspective

[3] Just avoiding the OpenCV's Java wrapper and program it in C will already entail an important speedup.

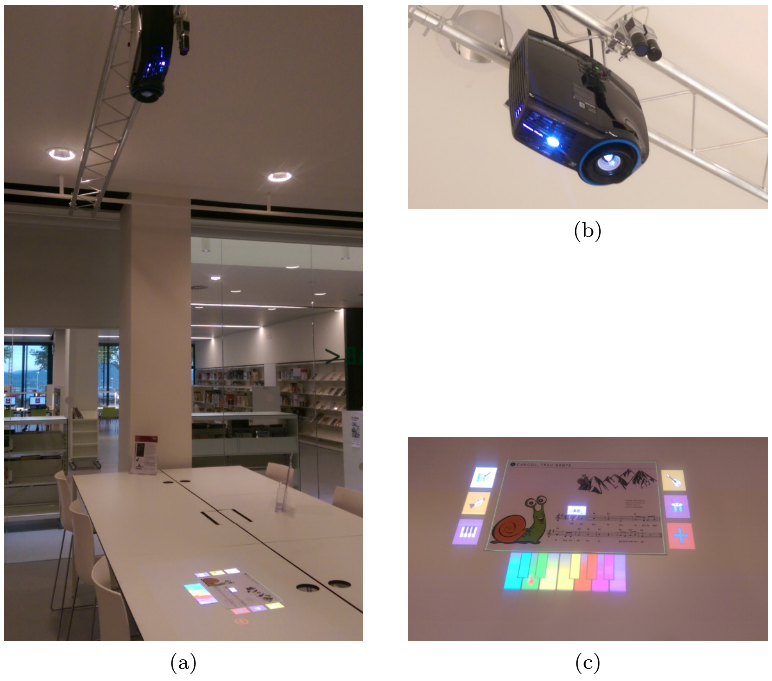


Fig. 13: my caption. (a) is .... (b) is .... (c) is ....

transform estimation. This benchmark provides a baseline for the particular use case of augmented documents on mobile devices. In order to conduct those experiments, we introduced an original and public dataset precisely ground-truthed consisting of a total of 21 048 frames. Results show that the performance yielded by the ORB feature descriptor is comparable with the more computationally demanding SIFT descriptor.

The mobile prototype is able to run the display in real time, with an adjustment of the augmented content position at a rate of 7 *fps*, thanks to a separation of the document matching process and the AR rendering process. We tested the prototype in real environments under different setups and with several parents and their children, obtaining a positive reactions.

Acknowledgements This work was supported by the Spanish project TIN2014-52072-P, by the People Programme (Marie Curie Actions) of the Seventh Framework Programme of the European Union (FP7/2007-2013) under REA grant agreement no. 600388, by the Agency of Competitiveness for Companies of the Government of Catalonia, ACCIO, by the CERCA Programme / Generalitat de Catalunya, and by the MOBIDEM project, part of the "Systematic Paris-Region" and "Images & Network" Clusters, funded by the French Government and its economic development agencies.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM – 50th anniversary issue: 1958 – 2008 **51**(1), 117–122 (2008)
2. Argelia, N.: How to include augmented reality in descriptive geometry teaching. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 250–256 (2015)
3. Bacca, J., Baldiris, S., Fabregat, R., Graf, S., Kinshuk: Augmented reality trends in education: a systematic review of research and applications. Journal of Educational Technology & Society **17**(4), 133 (2014)
4. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: SURF: Speeded up robust features. Computer Vision and Image Understanding **110**(3), 346–359 (2008)
5. Bin, A., Rohaya, D.: An interactive mobile augmented reality magical playbook: Learning number with the thirsty crow. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 123–130 (2013)
6. Burie, J., Chazalon, J., Coustaty, M., Eskenazi, S., Luqman, M., Mehri, M., Nayef, N., Ogier, J., Prum, S., Rusiñol, M.: ICDAR2015 competition on smartphone document capture and OCR (smartdoc). In: Proceedings of the 13th International Conference on Document Analysis and Recognition, pp. 1161–1165 (2015)
7. Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., Fua, P.: BRIEF: Computing a local binary descriptor very fast. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(7), 1281–1298 (2012)
8. Cascales, A., Pérez-López, D., Contero, M.: Study on parent's acceptance of the augmented reality use for preschool education. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, vol. 25, pp. 420–427 (2013)
9. Chazalon, J., Rusiñol, M., Ogier, J.: Improving document matching performance by local descriptor filtering. In: Proceedings of the 6th International Workshop on Camera Based Document Image Analysis, pp. 1216–1220 (2015)
10. Chazalon, J., Rusiñol, M., Ogier, J., Lladós, J.: A semi-automatic groundtruthing tool for mobile-captured document segmentation. In: Proceedings of the 13th International Conference on Document Analysis and Recognition, pp. 621–625 (2015)
11. Diaz, C., Hincapié, M., Moreno, G.: How the type of content in educative augmented reality application affects the learning experience. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 205–212 (2015)
12. Everingham, M., Gool, L.V., Williams, C., Winn, J., Zisserman, A.: The PASCAL visual object classes (VOC) challenge. International Journal on Computer Vision **88**(2), 303–338 (2010)
13. Fischler, M., Bolles, R.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
14. Goodwin, A., Green, R.: Key detection for a virtual piano teacher. In: Proceedings of the International Conference of Image and Vision Computing (2013)
15. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004)
16. Huand, F., Zhou, Y., Yu, Y., Wang, Z., Du, S.: Piano ar: A markerless augmented reality based piano teaching system. In: Proceedings of the International Conference on Intelligent Human-Machine Systems and Cybernetics, pp. 47–52 (2011)
17. Jain, R., Oard, D., Doermann, D.: Scalable ranked retrieval using document images. In: Proceedings of Document Recognition and Retrieval XXI (2013)
18. Karatzas, D., d'Andecy, V., Rusiñol, M., Chica, A., Vazquez, P.: Human-document interaction systems a new frontier for document image analysis. In: Proceedings of the 12th IAPR Workshop on Document Analysis Systems, pp. 369–374
19. Leutenegger, S., Chli, M., Siegwart, R.: BRISK: Binary robust invariant scalable keypoints. In: Proceedings of the International Conference on Computer Vision, pp. 2548–2555 (2011)
20. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)

21. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of Imaging Understanding Workshop, pp. 121–130 (1981)
22. Martínez, M., Diaz, F., Barroso, L., González, D., Antón, M.: Mobile serious game using augmented reality for supporting children's learning about animals. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 375–381 (2013)
23. Moraleda, J.: Large scalability in document image matching using text retrieval. Pattern Recognition Letters **33**(7), 863–871 (2012)
24. Moraleda, J., Hull, J.: Toward massive scalability in image matching. In: Proceedings of the International Conference on Pattern Recongition, pp. 3424–3427 (2010)
25. Muja, M., Lowe, D.: Fast approximate nearest neighbors with automatic algorithm configuration. In: Proceedings of the International Conference on Computer Vision Theory and Applications, pp. 331–340 (2009)
26. Nakai, T., Kise, K., Iwamura, M.: Use of affine invariants in locally likely arrangement hashing for camera-based document image retrieval. In: Proceedings of the International Workshop on Document Analysis Systems, pp. 541–552 (2006)
27. Nayef, N., Luqman, M., Prum, S., Eskenazi, S., Chazalon, J., Ogier, J.: SmartDoc-QA: A dataset for quality assessment of smartphone captured document images - single and multiple distortions. In: Proceedings of the International Conference on Document Analysis and Recognition (2015)
28. Pra, Y.D., Fontana, F., Tao, L.: Infrared vs. ultrasonic finger detection on a virtual piano keyboard. In: Proceedings of the ICMC (2014)
29. Quintero, E., Salinas, P., González, E., Ramírez, H.: Augmented reality app for calculus: A proposal for the development of spatial visualization. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 301–305 (2015)
30. Rohaya, D., Matcha, W., Sulaiman, S.: Fun learning with ar alphabet book for preschool children. In: Proceedings of the International Conference on Virtual and Augmented Reality in Education, pp. 211–219 (2013)
31. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: Proceedings of the International Conference on Computer Vision, pp. 2564–2571 (2011)
32. Rusiñol, M., Chazalon, J., Ogier, J., Lladós, J.: A comparative study of local detectors and descriptors for mobile document classification. In: Proceedings of the 13th International Conference on Document Analysis and Recognition, pp. 596–600 (2015)
33. Takeda, K., Kise, K., Iwamura, M.: Real-time document image retrieval for a 10 million pages database with a memory efficient and stability improved LLAH. In: Proceedings of the 11th International Conference on Document Analysis and Recognition, pp. 1054–1058 (2011)
34. Takeda, K., Kise, K., Iwamura, M.: Real-time document image retrieval on a smartphone. In: Proceedings of the International Workshop on Document Analysis Systems, pp. 225–229 (2012)
35. Telea, A.: An image inpainting technique based on the fast marching method. Journal of graphics tools **9**(1), 23–34 (2004)
36. Wu, C., Hsu, C., Lee, T., Smith, S.: A virtual reality keyboard with realistic haptic feedback in a fully immersive virtual environment. Virtual Reality pp. 1–11 (2016)
37. Yilmaz, R.M.: Educational magic toys developed with augmented reality technology for early childhood education. Computers in Human Behavior **54**, 240–248 (2016)