# Book-AR: Augmented Reality for Books on a Bookshelf

**Yerbol Aussat**
University of Waterloo
Waterloo, Canada
yaussat@uwaterloo.ca

## ABSTRACT

Retrieving information about books on a bookshelf by simply pointing a camera at them could make the library and bookstore browsing experience more convenient and effective. This paper presents Book-AR, an interactive real-time augmented reality (AR) system that augments books while they are on a bookshelf. Our system recognizes a desired book by its spine, and displays an AR menu with relevant book information, such as genre, ratings, prices and similar books. The user can interact with Book-AR by clicking on book spines to open and close the interactive menu, and swiping left and right to switch between pages of the menu. The paper discusses a prototype of the system and explains its implementation techniques.

## INTRODUCTION

When browsing through a library or a bookstore, we typically look through bookshelves trying to find a title that catches our eye. If a title seems interesting, we reach for the book, pull it out from a bookshelf, and inspect its front and back covers. If this information is not sufficient, we open a search engine on our smartphone, and insert the title of the book in hopes of finding the book's ratings and critiques. Moreover, typically we want to get the price of a book; some of us go a step further by comparing the prices from major vendors. If we are not looking for something specific, and just browsing bookshelves, we likely repeat this process several times.

In this project we design and implement an interactive system with a goal to make the bookstore and library browsing process simpler, faster and more convenient. Modern smartphones are equipped with high-resolution cameras and access to high-speed networks. This makes many computer vision-based interactive applications possible. Many visual search applications such as Google Goggles [9] now enable smartphones to retrieve information about products simply by snapping a photo. There is also a growing interest in applications that augment smartphone and tablet video streams with relevant information about the objects currently visible. Existing augmented reality applications use a smartphone's camera and other sensors to create virtual layers of digital information on top of various objects of interest seen in the phone's video stream [5].


**Figure 1. Interaction between the user and the BookAR system. (a) Camera's field of view; (b) Visual feedback that arises when a user covers the bottom of the book spine; (c, d, e) Pages of the AR menu; (f) Deselection of the book.**

We develop a prototype of an interactive real-time augmented reality system that can augment a book spine with a relevant book information. An interaction between a user and the Book-AR system is illustrated on Figure 1. First the user points a camera at the bookshelf so that the book spine of interest is in the camera's field of view. The user then can select the book spine by covering its lower part with a hand. The system then provides visual feedback by displaying a book's cover transparently in the AR fashion, as illustrated in Figure 1(b). If the lower part of a book spine is being covered for at least

Link to a demo: https://www.youtube.com/watch?v=UObxvmxuuC0

ACM ISBN .
DOI:

3 seconds, the book's status changes to "selected", and the AR image of the book cover becomes opaque, as shown in Figure 1(c). This is the first page of the AR menu. The menu of the current prototype of Book-AR has in total 3 pages that are shown on Figure 1(c-e). The second page contains information about the book's genre, ratings from literature review websites and prices from major vendors, whereas the third page suggests similar books. The book spine serves as an AR marker to anchor the menu. Therefore, the menu appears physically attached to the book spine. To flip through the menu pages, the user can move his hand right or left in front of the camera. Deselection of the book is done exactly in the same way as selection. To deselect the book and disable AR menu the user needs to hold his hand at the bottom of the book spine for 3 seconds, as illustrated in Figure 1(f).

The main contributions of this work are:

- A prototype of a robust interactive real-time augmented reality system that augments the book spine of the selected book with the relevant information.
- Implementation of a reliable book selection technique that employs SIFT features.
- Implementation of a reliable interactive technique for switching between AR menu's pages based on the coarse optical flow of a single pixel.

## RELATED WORK

One of the most important functionalities of Book-AR is book spine recognition. A number of previous studies applied computer vision techniques to tackle the problem of recognition of a book on a bookshelf by its spine.

Several authors propose marker-based approaches to recognize and track books on shelves [8, 1, 11, 13]. The main idea behind these approaches is that they use distinct markers placed on book spines to uniquely identify books. When a shelf is viewed through a camera, books can be easily recognized due to information encoded in the markers. Fiducial markers significantly simplify the task of identifying a book, and systems that use them are fairly accurate in controlled settings. However, such systems require markers to be placed on every book in a library or a bookstore, which might significantly increase the deployment expense. Therefore, this approach is not the best for our system.

In more recent systems book spines themselves are used for identifying and tracking books, and this is the goal of our system. In their work Lee et al. [6] proposed an image matching method that quantizes color images of spines into a limited number of color indices. The authors make simplifying assumptions that books are perfectly aligned with the vertical axis and each image is taken right in front of a bookshelf. Also, the system does not perform well on books of similar color.

Several studies examine recognizing the identity of books on a bookshelf by extracting their titles [15, 14, 16]. Quoc and Choi propose a similar system for book recognition, which performs book region extraction, book segmentation, and title extraction to identify a book [14]. The authors use optical character recognition (OCR) to read the book title from the segmented book spines. This approach is not robust against photometric and geometric distortions. Even though our system does not use OCR, we think that this technique might be complimentary to other book spine recognition methods. This paper further discusses OCR in the Discussion section.

A more powerful approach for book spine recognition is to use local image features such as SURF, SIFT, or FAST [10, 2, 3, 4, 16, 16]. Chen et al. [2, 3] have implemented a mobile book recognition and tracking system for generating an inventory of books by taking a picture of a bookshelf. The system achieves a robust automatic book recognition. First, it segments individual book spines, and extracts a set of visual features for each query spine. Next, the features are matched with a spine database, and recognized spine identifiers are displayed on the screen. The authors believe that spine segmentation is critical for accurate feature-based image matching, as it greatly reduces clutter from the perspective of any single spine. In terms of used computer vision techniques, our system is quite similar to this system by Chen et al. The difference is that we do not do spine segmentation before matching features, and there are several reasons for it. First, based on our experiments, using visual features alone is robust enough regardless of the background clutter. Second, spine segmentation itself is not very reliable. It is strongly dependent on lighting conditions, and therefore, provides an additional source of error. Finally, the goal of Book-AR is not only to recognize a presence of a book spine in an image, but also to detect its position in the image and perspective transformation with respect to a stored query book spine. Therefore, visual features of each query book spine need to be matched against the entire video frame.

In their follow-up work Chen et al. [4] implement an augmented reality system capable of real-time book spine recognition. The system determines a user's interest in a book by analyzing the motion of objects seen in the viewfinder, and initiates a query during each low-motion interval. Compared to the system reported in [2, 3], this system has a reduced recognition delay of 1 second, which still might be too long for interactive AR applications. Compared to previous works, recognition latency reduction is achieved by quickly selecting a query frame from viewfinder frames at the start of a low-motion interval. Just like Book-AR, the system by Chen et al. is an AR system. After the book is recognized, the viewfinder is augmented with a book cover image, prices from different vendors, average user rating, and other relevant information. At first glance, the description of this system sounds very similar to Book-AR. However, there are a few significant differences. First, Book-AR is interactive. With Book-AR a user can select a book spine or flip AR menu pages by simple hand gestures, whereas the system by Chen et al, does not support any input gestures. Second, in their system the augmented menu is just a standard 2-d window on a phone that is the same regardless of the phone's perspective. In turn, in our system AR graphics appears physically attached to the book spine, which provides the feeling of natural interaction.

## SYSTEM OVERVIEW

The current proof-of-concept prototype of Book-AR system is implemented and run on a Macbook laptop. Except for laptop, the only required piece of hardware is a web camera. The camera used in this work is Microsoft LifeCam HD-5000. From our experience, the best performance is achieved when auto-focus and auto-exposure are turned off.

For system implementation Python programming language and OpenCV computer vision library are used. In terms of functionality, system implementation can be divided into three parts: recognition and tracking of book spines, display of augmented reality graphics, and system interactions via hand gestures. All of these three functionalities are "per frame", meaning that book spines are recognized, AR graphics are displayed, and gestures are detected in every frame. These functionalities first have been implemented and tested independently, and only then combined into the system. Therefore, we start this section by discussing each of these parts individually. Then we conclude the section by discussing system workflow and describing how these parts work together as the Book-AR system.

### Book Spine Recognition and Tracking

Figure 1(a) shows a typical query image of a bookshelf where book spines need to be detected and recognized. Recognition of spines in such images is a challenging task. First, because of spines' small surface areas relatively few visual features can be extracted for robust recognition. Second, an image of a bookshelf typically contains several book spines that tend to act as a background clutter.

#### *Feature Extraction*

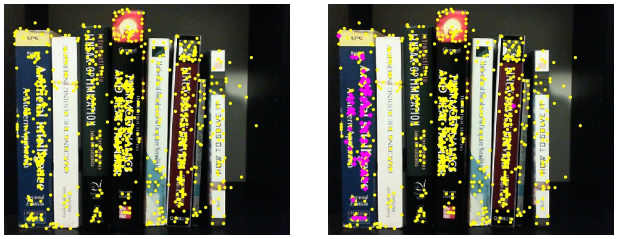
Our system's approach for book spine detection and recognition in a video frame is to match each video frame against a database of book spines. Specifically, first visual features are extracted from each video frame, as depicted in Figure 2(a). Then the video frame's features are matched against visual features of book spine images from the database. For augmented reality systems it is required that visual features are scale- and rotation-invariant. Also, since Book-AR is an interactive system, achieving low query latency is desired. At the same time, since book spines serve as anchors to an AR interactive menu, it is necessary that tracking and detection errors are low.

SIFT, SURF and ORB are the most widely used feature descriptors invariant to scale and rotation. SURF and ORB detectors are slightly faster compared to SIFT. However, SIFT has a better performance in terms of tracking and feature point drift [7]. After conducting a small experiment, it has been confirmed that SIFT detector is significantly more robust in situations when the number of extractable features is limited. It has also been found that the difference in speed between three feature detectors is not practically noticeable. Therefore, SIFT features are employed in the project. SIFT feature detector can be easily initialized in OpenCV with `cv2.xfeatures2d.SIFT_creat` function, and SIFT keypoints and descriptors can be extracted with `sift.detectAndCompute` function.

#### *Feature Matching*

After SIFT features are extracted from the video frame, Fast Library for Approximate Nearest Neighbors (FLANN) - based matcher is used to find matches between the video frame and each of the query book spines in the database. The process of finding correspondences between the video frame and a spine image can be formulated as a search for the nearest neighbor from one set of descriptors for every element of another set. FLANN matcher returns a list of correspondences between two sets of descriptors. In OpenCV the matches can be found with a `flann.knnMatch` function, where `flann` is a FLANN matcher object. However, there is a high chance that some matches are false positives. The technique used for removing false positives is called ratio test, which requires to use two nearest neighbors [12]. According to the ratio test, the match is good only if the distance ratio between the first and the second matches is smaller that a certain number, which is chosen to be 0.7 in the current prototype of the system. Figure 2(b) and Figure 3 illustrate good matches found by the FLANN matcher with the ratio rest employed to get rid of ouliers. If more than 15 good matches between the video frame and some book spine image are detected, the program concludes that this book spine is present in the current video frame. Otherwise, the matches are assumed to be a noise and ignored.

In order to track book spines, the program simply does book spine detection and recognition in each video frame. Matching video frame against many book spine images can be a quite slow process. In order to reduce latency, we also experimented with combining SIFT feature descriptors with sparse KLT optical flow. The idea of this technique is that KLT optical flow is used to track good SIFT keypoints, and the keypoints are recalculated with a certain periodicity. This approach turned out to be not suitable for our task, because swipe gestures move the tracked keypoints making them invalid. Moreover, this technique still lags every tenth video frame, when SIFT features are recalculated. The issue of latency is further discussed in the Discussion section.



(a) (b)

**Figure 2. (a) SIFT features, indicated with yellow dots, extracted from an image of a bookshelf. (b) Good matches between the bookshelf's image and one of the book spines are indicated with purple dots. FLANN matcher with k=2 and Lowe's ration test are used for robust feature matching.**

### Displaying AR Graphics

When a book spine is selected, the system displays an augmented reality menu with relevant book information, as illustrated on Figure 1(c-e). The book spine is used as an AR marker to anchor the menu. The goal is to display AR graphics

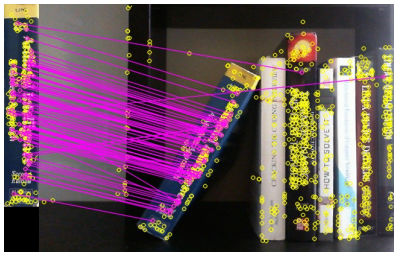


**Figure 3. Mapping of points from the plane of the book spine in the stored query image to the plane of the book spine in the video frame**

in a way that it looks like it is physically attached to the book spine from all camera perspectives.

*Homography Computation*

In order to accomplish this, a perspective transformation, or homography, is computed between book spine image and video frame. First, the book spine's reference surface is identified in the video frame and a set of valid matches are found, as discussed above. Once this is done, the homography can be calculated. The goal here is to find the transformation that maps points from the plane of the book spine in the stored query image to the plane of the book spine in the video frame. The mapping found by the FLANN matcher is illustrated in Figure 3. To find the correct perspective transformation, the `cv2.findHomography` function is used with the random sample consensus (RANSAC) method to probe different subsets of input points. Since it cannot be guaranteed that all the matches the FLANN matcher has found are valid matches we have to consider that there might be some false matches, or outliers. Therefore, we have to use an estimation method that is robust against outliers, such as RANSAC. RANSAC is an iterative algorithm used for model fitting in the with a large number of outliers.

*Image Rectification*

Perspective transformation can be used to transform a stored image of AR menu to the plane of a book spine in the video frame. In other words, each pixel of the stored AR menu's image is mapped to the corresponding point in video frame by application of computed homography matrix transformation. In OpenCV this can be easily achieved by using `cv2.warpPerspective` function. Figure 4 shows the result of applying homography transformation from Figure 3 to the AR menu.

*Transparent overlay of the AR menu*

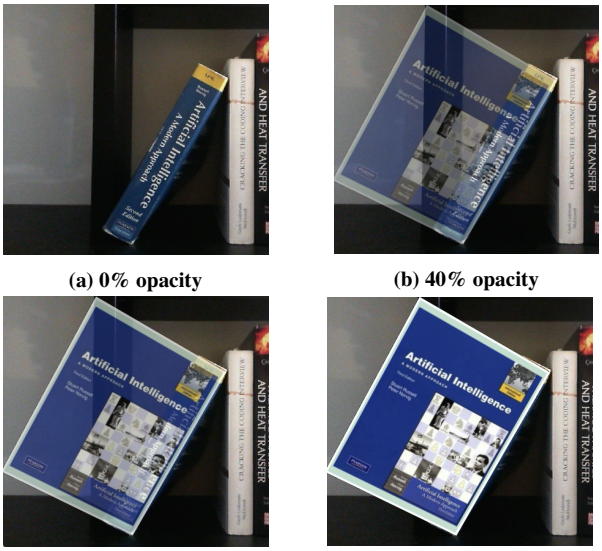
After the warped perspective of the AR menu is computed, it can be projected to the current video frame, so that the menu appears attached to the book spine. To achieve it, the warped AR image pixels need to be overlaid on top of the current video frame. We also want to project the first page of the menu transparently to provide a user with visual feedback during book selection, as discussed above.

For overlaying the menu transparently OpenCV's `cv2.addWeighted` function can be used. In addition to two images to be overlaid, this function takes a parameter alpha as an input. This parameter specifies the level of


**(a) Stored image of the AR menu (b) Warped image of the AR menu**

**Figure 4. Homography transformation that maps a stored image of the AR menu to the plane of the book spine in the current video frame**

transparency of the overlay, and its effect is shown on Figure 5. The closer alpha is to 1.0, the more opaque the overlay is. Similarly, the closer alpha is to 0.0, the more transparent the overlay appears.




**(a) 0% opacity (b) 40% opacity**

**(c) 70% opacity (d) 100% opacity**

**Figure 5. Transparent overlay of the AR menu**

Calculation of a perspective transformation matrix and projection of the AR menu to the video frame takes place in every frame.

## Interaction Techniques

The set of interactions between a user and the Book-AR system includes selecting a book of interest, switching between pages in AR menu, and closing the menu. To select or deselect a book, the user needs to touch the corresponding book spine, as shown in Figure 1(b) and 1(f) respectively. For flipping through AR menu's pages, the user needs to swipe right or left.

Fingertip tracking techniques are widely used for input gesture implementation in computer vision based applications. The most popular fingertip tracking techniques are based on

convexity defects and k-curvatures of hand contours. However, these techniques are not robust for many applications due to their dependence on skin color detection for extracting contours. Isolating skin regions reliably from the background would be very challenging in our application because a bookshelf with book spines of various colors makes a very cluttered background. Next, we propose techniques for reliable implementation of book selection and swipe gestures.

*Book selection*

Book selection input is used for two purposes: selecting a book of interest, and closing an AR menu. For a user to select a book, he or she needs to cover the bottom of the corresponding book spine. If the system records that the bottom of the book spine is covered for more than 2 consecutive video frames, or roughly 1 second, the system provides visual feedback to a user, as illustrated in Figure 1(b). After the book spine's bottom is covered for 6 consecutive frames, or roughly 3 seconds, book's status changes to "Selected" and AR menu pops up. The visual feedback is 40% opaque, whereas AR menu is 100% opaque, as depicted on Figure 5(b) and Figure 5(d) respectively.

The program concludes that the book spine is present in a current video frame if it detects more than 15 good matches between the video frame and book spine images. Otherwise, the matches are assumed to be a noise and ignored.

In order to implement a book selection input, Book-AR uses SIFT feature matches between a stored book spine's image and the current video frame. As mentioned above, if more than 15 good matches between the video frame and a book spine image are detected, the program concludes that this book spine is present in the current video frame. Next, it counts a fraction of good matches found in the lowest quarter of the book spine.

Figure 6(a) shows a video frame, where a book spine is not selected, whereas Figure 6(b) shows a frame with a selected spine. On these figures, good matches in the lowest quarter of the book spine are indicated as red circles, and all other good matches are indicated as yellow circles. It is quite evident that when a book spine is not selected, many good matches are found in its lowest quarter. At the same time, when a book spine is selected, only a few good matches are present in the spine's lowest quarter. Hence, to decide whether the book spine is selected, the following simple heuristic is used. If less than one tenth of all good matches is located in the lowest quarter of the book spine, the spine selection is registered.

Computing homography transformation between query book spine image and a book spine in the video frame is the initial step to calculating a number of matches in the lowest quarter of the book spine. Next, using OpenCVâĂŹs cv2.warpPerspective function, we warp perspective of the book spine in the video frame to the coordinate system of query book spine. These warped images are depicted on top right corners of Figures 6(a) and 6(b). As a result of this transformation, the book spine from the video frame is brought onto the frontal plane, and positioned in the same orientation as the stored query book spine. After this perspective transformation is done, the number of good matches that lay in the lowest quarter of the book spine can be counted directly, by thresholding y-coordinate of the warped spine image.


(a) (b)

**Figure 6. Book spine selection technique based on SIFT features. (a) Book spine is not selected. Many good matches are found in the spine's lowest quarter. (b) Book spine is selected. Only a few good matches are found in the spine's lowest quarter.**

*Swipe gestures*

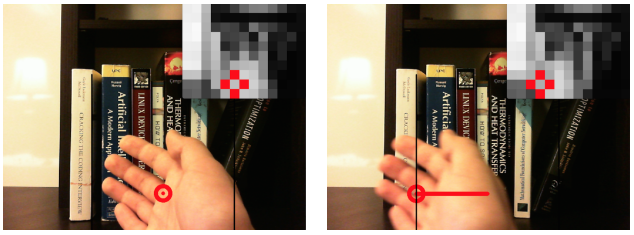
Swipe input gestures are used to switch back and forth between pages of AR menu. Book-AR uses global optical flow to implement swipe gestures. Hand swipes are typically quick motions over relatively long distances. Regular optical flow methods are based on a small motion assumption, therefore they perform poorly if the object being tracked moves over a big number of pixels quickly. Coarse optical flow is often used to solve this problem. The main idea of this technique is to reduce resolution of the video frame, so that quick motions over long distances are translated into motions over a few pixels in a new frame of a reduced size. As a result, only coarse motions are detected. In Book-AR system the resolution of original frame is reduced from (480, 640) to (10, 13). At the top right corner of Figures 7(a) and 7(b), the video frames of reduced resolution are shown. These images demonstrate that reduced frames are sufficient to detect coarse hand motion in the camera's field of view.


(a) (b)

**Figure 7. Coarse optical flow: (a) A hand is not in the camera's field of view; (b) A hand is in the camera's field of view.**

After several experiments with coarse optical flow, it was found that tracking optical flow of a single pixel at a fixed coordinate is sufficient for reliable detection of coarse swipe gestures. A pixel with coordinates (6, 7) in the coarse frame's coordinate system has been chosen for tracking. This pixel is

marked by red circles in Figure 8, both in original and coarse video frames. Figure 8 illustrates how swipe detection problem is modeled as a single pixel tracking problem. Whenever the magnitude of the displacement vector's horizontal component exceeds a fixed threshold, a swipe is registered. The direction of swipe is defined by the sign of displacement vector's horizontal component. In other words, whenever the red circle on Figure 8 crosses right or left black vertical lines, the system registers right or left swipe respectively.



**(a) No swipes detected.** **(b) Left swipe detected**

**Figure 8. Swipe detection problem represented as the single pixel tracking problem. Whenever the red circle crosses right or left black vertical lines, the system registers right or left swipe respectively.**

In order to compute global optical flow of the pixel, OpenCV's `cv2.calcOpticalFlowFarneback` function is used. It takes two consecutive coarse video frames as inputs, and returns a matrix of pixel displacements.

### System Workflow

So far, we have discussed each of Book-AR's functionalities individually. In this subsection we consider the system as a whole, and explain how its different parts work together. The high level system workflow is illustrated in Figure 9.

The program starts with offline system initialization, which consists of two main parts. First part is extraction of scale- and rotation-invariant SIFT features from each query spine in the database. Second part includes import of images of all AR menu pages for each book in the database. In addition, camera initialization takes place at this step. Current prototype operates at resolution of (480, 640).

Algorithm extracts SIFT features from every new video frame for further book spine recognition and tracking, as discussed above. Next, depending on whether the book is selected, the system enters one of two modes. This is indicated by the "Book spine selected?" decision block on the system flow diagram on Figure 9.

If the book is not selected the program enters a book spine selection mode. In this mode the video frame's features are matched against the features of every query spine image from the database. Next, it is checked if any of the book spines are selected in the current frame, as described in the "Interaction Techniques" subsection. If a book spine is selected in 6 consecutive frames, the status of the system becomes "Spine selected". In the next frame after the "Book spine selected?" decision block the system enters a book spine augmentation mode.

In a book spine augmentation mode exactly one of the book spines has a "selected" status. In this mode, video frame's features are matched only against the selected book spine's features. The system then uses good matches to augment selected spine with an AR menu. Current AR menu page is stored in a global variable, which is reset to 0 every time a spine gets deselected. After AR graphics is projected onto the spine, the system checks if any of input gestures have been registered in the current frame.

If right or left swipe is registered, the program respectively increments or decrements the current AR menu page's variable. As a result, in the next video frame the book spine is augmented with the corresponding new AR menu page. It should be noted that during a single hand swipe the pixel whose optical flow is being tracked might get triggered more than once, because the hand is represented by multiple pixels in the coarse video frame, as can be seen in Figure 7(b). To improve robustness of the swipe gestures, the swipe detection is disabled in the following 5 frames if a swipe is detected in the current frame. This simple condition allows to avoid false positive swipe detection.

Book spine deselection mechanism is exactly the same as the one of a book spine selection. If the book spine that is currently being augmented with an AR menu is selected in 6 consecutive frames (as described in the "Book selection" subsection), the status of the system becomes "Spine not selected", and in the next frame after the "Book spine selected?" decision block the system enters a book spine selection mode.

## DISCUSSION AND FUTURE WORK

While detailed user experiments have not been performed, informal user feedback from students collected during the project demo has been very positive. The descriptors given to the Book-AR system during the demo are "neat", "cool", "awesome". All students found the swipes and book selection input gestures very intuitive. One of the things many people particularly liked about the system was that the AR menus appear physically attached to a book spine from all camera perspectives. People see the potential practical value of the system in libraries and bookstores. Also, several students noted that the system would be particularly convenient if used with an AR headset rather than with a web camera or a mobile phone.

In terms of technical performance, Book-AR is reliable, with all desired functionalities working as intended. The system is robust to different lighting conditions, because all individual techniques perform well in a wide range of differently illuminated environments. In a book spine selection mode the system successfully detects, recognizes and tracks all book spines in the camera's field of view from a wide range of perspectives. Because Book-AR employs SIFT features that are based on difference of Gaussians, spine detection is robust to rotation and change in scale. In a book spine augmentation mode the system is capable of successfully overlaying AR menu onto a book spine. It should be noted that AR menu is very stable if the camera's orientation is perpendicular to a book spine's surface. Even though no formal tests on the range of acceptable angles between camera orientation and

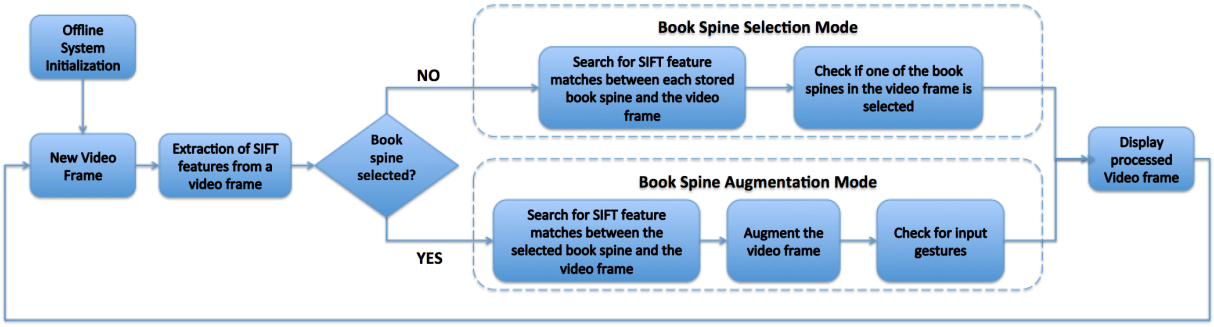



**Figure 9. System flow diagram**

the book spine's surface have been done, our initial informal tests showed that this range is sufficiently wide. Book-AR works reliably at most angles from which a person would read text from flat surfaces. One of the reasons of such a robustness is the RANSAC method that probes different subsets of input points, making sure that bad matches are not used for homography computation.

Our proposed interaction techniques perform well, despite the highly cluttered background. Based on initial tests, the SIFT features - based book selection technique is overall quite reliable. It is robust to the book spine's scale and tilt angle. However, this technique has some shortages. As mentioned in the previous section, spine selection is registered when less than one tenth of all good features is located in the lowest quarter of a book spine. Therefore, this technique relies on the assumption that good features are evenly distributed along book spines. If most of good features are concentrated on one certain part of a book spine, this technique might perform poorly. Besides, current implementation of Book-AR always treats a specific end of book spine as a bottom, depending on the orientation of the query spine image. Only this end has to be covered in order to trigger book selection. This problem can be easily fixed, by marking the spine's end that has a lower vertical coordinate in the original video frame as a bottom.

The initial tests demonstrate that swipe gestures, based on a single pixel's coarse optical flow, are simple, intuitive and robust. The biggest problem with current implementation of the optical flow - based swipe gestures is unintentional triggering of swipe inputs when camera moves quickly. One way to solve this problem is to attach accelerometers to the camera so that the system could differentiate between the motion of the camera and the motion of objects in front of the camera. It should be pointed out that accelerometers are already available on smartphones and other devices, where the final version of the system would be run.

There are a few other system limitations that are worth mentioning. In a book spine selection mode each video frame is matched against the database of all book spines. Our proof-of-concept prototype was tested on a database consisting of only five books, which makes the feature matching process quick. However, such an approach is not very scalable for book spine databases of much larger sizes. Improving scalability of the system is one of the most important future steps. One possible way to make the system more scalable is to first use optical character recognition (OCR) in order to shortlist the book spine candidates. Then feature matching could be done on a smaller list of book spines. It would also be interesting to explore deep learning - based approaches for detecting and recognizing book spines, since deep neural networks are known to have good scalability.

There are also two system limitations that are harder to solve. First, SIFT features are not fast enough to support smooth real-time functionality. Currently the system operates at 2 frames per second. Even if this speed might be sufficient for systems such as Book-AR, the latency is still noticeable. Second, some of the book spines are too narrow, which makes it difficult to extract good SIFT features from them and to use them as AR markers.

For Book-AR to have a real practical value, a smarphone version of the system should be implemented. The augmentation experience can also be further extended to smart glasses, which would give a natural interaction experience.

Finally, a formal user and technical evaluations should be conducted in the future to further understand advantages and disadvantages of the system.

## CONCLUSION

In this paper we presented Book-AR, an interactive system that augments book spines with AR menu that contains information about the book. Our system has a great potential of making library and bookstore browsing experiences simpler, faster and more convenient. We have described computer vision techniques employed in the system, and proposed reliable interactive techniques for book selection and swipe gestures. Finally, strengths and limitations of the system have been discussed, and possible directions of future work have been suggested.

## REFERENCES

1. B. Brinkman. 2012. "ShelvAR". Project Website. (2012).
2. D. Chen, S. Tsai, C.-H. Hsu, K.-Y. Kim, J. P. Singh, and B. Girod. 2010a. Building book inventories using smartphones. In *ACM Multimedia (MM)*.

3. D. Chen, S. Tsai, C.-H. Hsu, K.-Y. Kim, J. P. Singh, and B. Girod. 2010b. Low-cost asset tracking using location-aware camera phones. In *SPIE Workshop on Applications of Digital Image Processing (ADIP)*.
4. D. Chen, S. Tsai, C.-H. Hsu, J. P. Singh, and B. Girod. 2011. Mobile augmented reality for books on a shelf. In *IEEE Workshop on Visual Content Identification and Search (VCIDS)*.
5. D.Chen, S.Tsai, R.Vedantham, R.Grzeszczuk, and B.Girod. 2009. Streaming mobile augmented reality on mobile phones. In *International Symposium on Mixed and Augmented Reality (ISMAR)*. ISMAR, 181–182.
6. D.Lee, Y.Chang, J.Archibald, and C.Pitzak. 2008. Matchingbook-spine images for library shelf-reading process automation.. In *In Proc. IEEE International Conference on Automation Science and Engineering (CASEâĂŹ08)*. 738–743.
7. Karami E., Prasad S., and Shehata M. 2011. Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images. In *In Proceedings of the 2nd Augmented Human International Conference*.
8. Reitmayr G. and Schmalstieg D. 2003. Location based applications for mobile augmented reality.. In *Proceedings of the Fourth Australasian Conference on User Interfaces*.
9. Google. 2010. "Google Goggles: use pictures to search the web". (2010). `http://www.google.com/mobile/goggles`
10. Matsushita K., Iwai D., and Sato K. 2011. Interactive bookshelf surface for in situ book searching and storing support.. In *In Proceedings of the 2nd Augmented Human International Conference*.
11. M. Loechtefeld, S. Gehring, J. Schoening, and A. Krueger. 2010. Shelftorchlight: Augmenting a shelf using a camera projector unit.. In . *UBIProjection 2010 - Workshop on Personal Projection*.
12. D. G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* 60, 2 (November 2004), 91–110.
13. Malhotra N., Singh A., DivyaKrishna J., Saini K, and Gupta N. 2014. Context-aware Library Management System using Augmented Reality. *International Journal of Electronic and Electrical Engineering* 7, 9 (2014), 923–929.
14. N. Quoc and W. Choi. 2009. A framework for recognition books on bookshelves.. In *In Proc. International Conference on Intelligent Computing (ICICâĂŹ09)*. 386–395.
15. Eiji T., Seiichi U., and Hiroaki S. 2004. Block Boundary Detection and Title Extraction for Automatic Bookshelf Inspection.. In *In: 10th Korea-Japan Joint Workshop on Frontiers of Computer Vision*.
16. S. Tsai, D. Chen, H. Chen, C.-H. Hsu, K.-H. Kim, J. P. Singh, and B. Girod. 2011. Combining image and text features: a hybrid approach to mobile book spine recognition. In *ACM Multimedia (MM)*.