FRAME-LEVEL MATCHING OF NEAR DUPLICATE VIDEOS BASED ON TERNARY FRAME DESCRIPTOR AND ITERATIVE REFINEMENT

Kyung-Rae Kim, Won-Dong Jang, and Chang-Su Kim

School of Electrical Engineering, Korea University, Seoul, Korea
E-mails: {krkim, wdjang}@mcl.korea.ac.kr, changsukim@korea.ac.kr

ABSTRACT

A frame-level video matching algorithm, which achieves dense frame matching between near-duplicate videos, is proposed in this work. First, we propose a ternary frame descriptor for the near-duplicate video matching. The ternary descriptor partitions a frame into patches and uses ternary digits to represent relations between pairs of patches. Second, we formulate the frame-level matching problem as the minimization of a cost function, which consists of matching costs and adaptive unmatching costs. We develop an iterative refinement scheme that converges to a local minimum of the cost function. The iterative scheme performs competitively with the global optimization techniques while demands a significantly lower computational complexity. Experimental results show that the proposed algorithm achieves effective frame description and efficient frame matching of near duplicate videos.

***Index Terms*—** Near-duplicate video detection, frame-level video matching, ternary frame descriptor, and iterative refinement.

1. INTRODUCTION

With the popularization of video-sharing sites, such as YouTube [1] and Vimeo [2], an enormous amount of videos are available on the Internet. A typical video database contains near-duplicate videos, which have the same contents but different modifications, *e.g.* subtitle insertion, contrast enhancement, and cropping. Near-duplicate videos induce redundancy and inefficiency in video retrieval and database management. Attempts have been made to identify near-duplicate videos by measuring similarities between videos in [3, 4]. However, since these techniques directly compare video-level descriptors and neglect temporal relations of video frames, they are vulnerable to trimming and compilation. To achieve more precise and reliable video similarity measurement, many frame-level video sequence matching techniques have been developed [5–11].

Chen and Stentiford [5] shifted a video temporally and compared it to the other video to detect a near-duplicate segment. Their algorithm is only applicable to videos that have the same frame rate and share a single segment. Tan and Triggs [6] constructed a graph based on the visual similarity and the temporal consistency, and then detected partial near-duplicates by solving the network flow problem. Chen *et al*. [7] defined a graph whose nodes and edges represent key frames and their similarities, respectively. They extracted dense subgraphs as near-duplicate segments. Chiu *et al*. [8] detected near-duplicate segments by applying the Hough transform to the similarity matrix of two videos. However, the main objective of these techniques [5–8] is to determine only the presence of near-duplicate segments. On the other hand, in [9–11], more precise frame-level matching algorithms have been developed based on dynamic programming, which obtain the optimal solutions by minimizing the Levenshtein distance between videos. However, they are computationally complex and require a large memory space.

Frame description is also important to accomplish reliable video sequence matching. Global color descriptors [10, 12] are efficient to compute, but susceptible to illumination variations and image quality degradation. Local descriptors [13–16] are more robust against image rotation, cropping, and illumination variations. However, the local descriptors are sensitive to subtitles and logos, inserted into near-duplicate videos. Recently, patch-based frame descriptors have been developed to integrate local color information. Yeh and Cheng [11] constructed a weighted graph of local patches and used it to extract the spatial correlation descriptor. Chen *et al*. [7] partitioned a frame into rings and described them using the histogram of oriented gradients and relative mean intensities. While [11] and [7] provide promising performances, their floating-point descriptions incur computational burdens. Inspired by the local binary pattern [17], Shang *et al*. [18] proposed cascading binary comparison results of patches. The binary descriptor demands very low computational complexity, but provides mediocre performance because of coarse discretization.

In this paper, we present a novel frame descriptor for efficient near-duplicate video matching, called the ternary frame descriptor. As a patch-based descriptor, the proposed descriptor represents the order relations of patch intensities in ternary digits. To compute the similarity of the ternary descriptors of two frames in a fast manner, we transform the ternary digits into bits and employ the Hamming distance. Moreover, we propose a frame-level matching algorithm to obtain dense frame matching between near-duplicate videos. We first formulate a cost function for the matching, which consists of matching costs and adaptive unmatching costs. Then, we develop an iterative refinement scheme, which is guaranteed to converge to a local minimum of the cost function. Experimental results confirm that the proposed algorithm achieves effective frame description and efficient frame matching of near-duplicate videos.

The paper is organized as follows. Section 2 describes the ternary frame descriptor. Section 3 proposes the frame matching algorithm based on the iterative refinement. Section 4 discusses experimental results. Finally, Section 5 concludes this work.

2. TERNARY FRAME DESCRIPTOR

This section proposes an efficient frame descriptor, which is suitable for the frame matching between near-duplicate videos.

There are various types of geometric near-duplication, including rotation, flipping, and flopping. Among them, video flopping, which

This work was supported partly by the National Research Foundation of Korea(NRF) grant funded by the Ministry of Science, ICT & Future Planning (MSIP) (No. 2009-0083495), and partly by Samsung Electronics Co., Ltd.

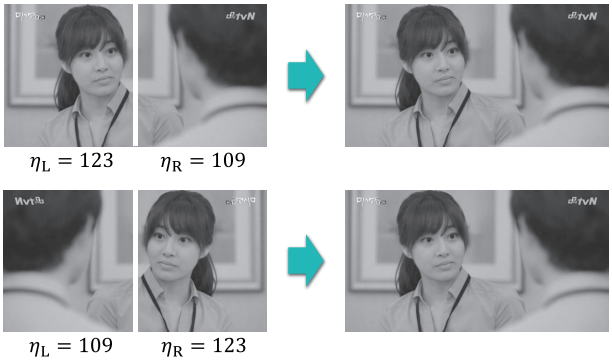


Fig. 1. Examples of the frame pre-flopping: each frame is flopped so that the average luminance of the left half frame is higher than that of the right one.

performs the mirror-reversal of a frame across the vertical axis, is the most frequent type. Hence, we develop a flopping-invariant descriptor by simply performing the frame flopping in a pre-determined manner. Specifically, we bisect an input frame vertically. Let η_L and η_R denote the average luminance intensities of the left and right half frames. If η_R is higher than η_L, we flop the frame across the vertical axis. Otherwise, we preserve the original frame. Fig. 1 illustrates the frame pre-flopping. Since the frame pre-flopping is applied to each frame, the proposed frame descriptor can be used to match partially flopped videos as well.

After the pre-flopping, we divide the frame into 4×4 patches and compute the average luminance intensity of each patch, as shown in Fig. 2. Let η_i denote the intensity of patch i. We then determine ternary digits by comparing the intensities of patches. Specifically, we obtain the ternary digit T_{ij} for patches i and j by

$$T_{ij} = \begin{cases} 0 & \text{if } \eta_i - \eta_j < -\tau, \\ 1 & \text{if } |\eta_i - \eta_j| \leq \tau, \\ 2 & \text{if } \eta_i - \eta_j > \tau, \end{cases} \quad (1)$$

where τ is a threshold that is fixed to 20. We compute the ternary digit for every possible pair of patches to obtain the frame descriptor $\mathbf{T} = \{T_{ij} : (i,j) \in \mathcal{N}\}$, where $\mathcal{N} = \{(1,2),(1,3),...,(15,16)\}$. Thus, the proposed ternary frame descriptor has the dimension of $120 = \binom{16}{2}$. It is noted that the ternary description of patch pairs was also adopted in the local ternary pattern for face recognition [19].

Let $\mathbf{T}(x)$ and $\mathbf{T}(y)$ denote the ternary descriptors of two frames x and y, respectively. We may measure their distance using the l_2-norm straightforwardly. However, for faster computation, we first convert each ternary digit T_{ij} into two bits by

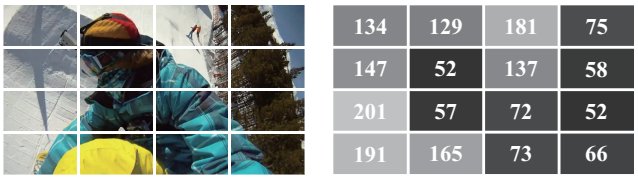
$$\tilde{T}_{ij} = \begin{cases} 00 & \text{if } T_{ij} = 0, \\ 01 & \text{if } T_{ij} = 1, \\ 11 & \text{if } T_{ij} = 2. \end{cases} \quad (2)$$

Then, we compute the Hamming distance to measure the difference between the descriptors $\mathbf{T}(x)$ and $\mathbf{T}(y)$ via

$$d_H(\mathbf{T}(x), \mathbf{T}(y)) = \sum_{(i,j)\in\mathcal{N}} \tilde{T}_{ij}(x) \oplus \tilde{T}_{ij}(y), \quad (3)$$

where $\oplus$ is the exclusive-or operator. The bitwise computation is reasonable, since the distance between ternary digits 0 and 2 is larger than that between 0 and 1 or that between 1 and 2.

The proposed ternary frame descriptor is a global frame feature that represents the order relation of patch intensities. It is hence more robust to various near-duplication processes, such as tone and color changes, than local features are. Moreover, in case of binary descriptors, a slight color change may switch a binary feature from 0 to 1 or from 1 to 0. In contrast, the ternary descriptor is designed to be more robust by employing the threshold τ in (1).



(a) (b)

Fig. 2. (a) Partitioning of a frame into 4×4 patches, and (b) the average luminance intensities of those patches.

3. FRAME MATCHING

We determine frame-by-frame correspondences between two near-duplicate videos. To reduce the complexity of the correspondence matching, we uniformly sample one frame per second from each video. Let $\mathcal{X} = \{x_1, x_2, ..., x_M\}$ and $\mathcal{Y} = \{y_1, y_2, ..., y_N\}$ denote the sets of the sampled frames from the two videos, respectively.

The correspondence matching can be formulated as an optimization problem. Then, dynamic programming can be employed to solve the problem [9–11], but it requires huge computational and memory complexities. To reduce these complexities, we propose an iterative frame matching algorithm that attempts to minimize the cost function robustly and rapidly.

3.1. Optimization Problem

For each frame $x_m \in \mathcal{X}$, we search the corresponding frame $y_n \in \mathcal{Y}$, whose index n is indicated by a matching function λ. In other words, a matching function λ is defined as

$$\lambda(m) = \begin{cases} n & \text{if } x_m \text{ is matched to } y_n, \\ 0 & \text{if } x_m \text{ is unmatched.} \end{cases} \quad (4)$$

The optimal matching function λ^* can be obtained by minimizing a cost function, given by

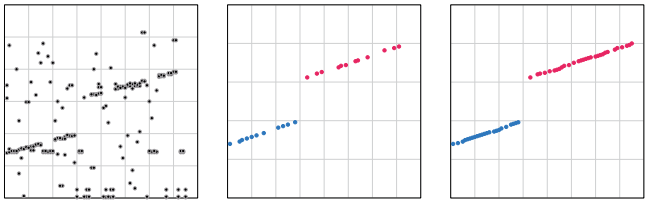
$$\lambda^* = \arg\min_{\lambda} \left(\sum_{m:\lambda(m)>0} d(m, \lambda(m)) + \sum_{m:\lambda(m)=0} c(m) \right), \quad (5)$$

where $d(m, \lambda(m))$ is the matching cost and $c(m)$ is the unmatching cost. As the matching cost $d(m, \lambda(m))$, we use the Hamming distance between the ternary frame descriptors of x_m and $y_{\lambda(m)}$,

$$d(m, \lambda(m)) = d_H(\mathbf{T}(x_m), \mathbf{T}(y_{\lambda(m)})). \quad (6)$$

The unmatching cost $c(m)$ imposes a penalty for an unmatched frame, which is necessary for avoiding the trivial solution that all frames are unmatched. Notice that x_m and y_n can be matched by the optimal matching function λ^*, only if $d(m,n) < c(m)$.

It is hard to determine a fixed unmatching cost that is appropriate for various near-duplication types. Suppose that two videos are visually quite different but contain the same contents. Since most matching candidates yield high costs, we need an even higher unmatching cost. In contrast, if two videos are almost identical, we require a low unmatching cost. Therefore, we determine the unmatching cost adaptively according to video contents.



(a) (b) (c)

Fig. 3. An example of the frame matching: (a) the best κ frames with the smallest matching costs for each frame in $\mathcal{X}$ in the 2D matching grid, (b) the initial matching functions, (c) the optimized matching functions after the iterative refinement.

For frame x_m, we calculate its matching costs to all frames in $\mathcal{Y}$ and obtain $\mathcal{D}_m = \{d(m,1), d(m,2), ..., d(m,N)\}$. Then, we sort the costs in $\mathcal{D}_m$ in the increasing order and denote the sorted indices as $i_1, ..., i_N$. We find the maximally increasing index by

$$\delta^* = \arg \max_{\delta \in \{2,\ldots,N\}} \left(d(m, i_\delta) - d(m, i_{\delta-1}) \right). \tag{7}$$

We then define the adaptive unmatching cost $c(m)$ for frame x_m as the average of the two consecutive matching costs, causing the maximal increasement, *i.e.*

$$c(m) = \frac{d(m, i_{\delta^*}) + d(m, i_{\delta^*-1})}{2}. \tag{8}$$

3.2. Initial Matching

We obtain initial matching functions between $\mathcal{X}$ and $\mathcal{Y}$. For each frame in $\mathcal{X}$, we find the best κ frames in $\mathcal{Y}$ to yield the smallest matching costs, where κ is set to 3 empirically. We construct a 2-dimensional matching grid, in which the x-axis and the y-axis represent the indices of the frames in $\mathcal{X}$ and $\mathcal{Y}$. As in Fig. 3(a), we plot the points, corresponding to the pairs of matching frames, on the matching grid. Matching subsequences tend to form lines in the grid. Therefore, we detect lines using the Hough transform [20]. Consequently, we have the initial matching functions $(\lambda_1^{\text{Init}}, ..., \lambda_Z^{\text{Init}})$, the angles $(\theta_1, ..., \theta_Z)$ and y-intercepts $(\rho_1, ..., \rho_Z)$ of detected lines, where Z is the number of the detected lines.

Fig. 3(b) shows initial matching results, in which many frames are unmatched. Since frame rates and sampled frames of videos may be different, it is hard to find dense matching at once using the Hough transform. We hence iteratively refine matching functions to achieve dense matching, as described in Section 3.3.

3.3. Iterative Matching Refinement

Let us consider a frame, x_m, which is located within the x-axis range of line z. To obtain its matching result $\lambda_z(m)$, we estimate the initial matching vector

$$\alpha_z(m) = m \cdot (\tan\theta_z - 1) + \rho_z, \tag{9}$$

where $\tan\theta_z$ is the ratio of the frame rates of $\mathcal{Y}$ and $\mathcal{X}$. To reduce the initial matching error and alleviate the effect of sampling errors, we expand the matching vector $\alpha_z(m)$ to its neighbors up to β_z displacements. Specifically, we set β_z by

$$\beta_z = \begin{cases} \lfloor \tan\theta_z \rceil & \text{if } \theta_z \geq \frac{\pi}{4}, \\ \lfloor \frac{1}{\tan\theta_z} \rceil & \text{otherwise,} \end{cases} \tag{10}$$

where $\lfloor \cdot \rceil$ is the round-off operator. Then, we find n^* that minimizes $d(m,n)$ by

$$n^* = \arg \min_{m+\alpha_z(m)-\beta_z \leq n \leq m+\alpha_z(m)+\beta_z} d(m,n). \tag{11}$$

As mentioned previously, our objective is to minimize the cost function in (5), and thus the matching should be allowed only if the matching cost is smaller than the unmatching cost. Thus, we refine the matching function by

$$\lambda_z(m) = \begin{cases} n^* & \text{if } d(m,n^*) < c(m), \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

This procedure is repeated until

$$\sum_{z=1}^{Z} \sum_{m} |\lambda_z^{(t)}(m) - \lambda_z^{(t-1)}(m)| = 0, \tag{13}$$

where t is an iteration index, and the inner summation is over the x-range of line z. Finally, we obtain the refined matching functions $\lambda_1^*, ..., \lambda_Z^*$, and combine them to obtain the final matching function λ^*. Fig. 3(c) shows the final matching function. Notice that the proposed refinement scheme decreases the cost function in (5) monotonically at each iteration. Thus, it is guaranteed that the iterative refinement converges to a local minimum of the cost function. Moreover, as shown in Section 4, the proposed algorithm provides a solution, which is almost as efficient as the globally optimal solution of the dynamic programming, at a much lower complexity.

4. EXPERIMENTAL RESULTS

We collect the test database of video clips from video-sharing websites YouTube [1], Vimeo [2], and Soku [21]. The average length of these test videos is about three minutes, and their spatial resolutions vary from 320×240 to 640×480. The database contains various near-duplication types, including subtitle/logo, contrast enhancement, lighting, cropping, resolution variation, and flopping.

We preprocess the videos to remove borders. Border insertion is one of the most frequent near-duplication types, but its removal is relatively easy. We eliminate borders by inspecting pixel positions whose luminance intensities are constant throughout the sequence.

4.1. Efficacy of Ternary Frame Descriptor

To evaluate the efficacy of the proposed ternary frame descriptor, we form 100 test sets, each of which consists of 5 near-duplicate videos. From each video, 20 frames are sampled. Thus, 10,000 frames are used in total. We match frames with threshold adjustments. Note that near-duplicate frames should be matched to each other. At each threshold, the precision and recall rates are computed, which are defined as

$$\text{Precision} = \frac{\text{Number of true detected matches}}{\text{Number of detected matches}}, \tag{14}$$

$$\text{Recall} = \frac{\text{Number of true detected matches}}{\text{Number of true matches}}. \tag{15}$$

We compare the proposed ternary frame descriptor with the Yeh and Cheng's global descriptor [11], the binary descriptor, and the RGB color histogram. In the binary descriptor, binary relations, instead of ternary relations, are used for the relative description of patches. In the RGB color histogram, each color channel is quantized into 16 levels. Thus, the histogram is 48-dimensional. For

(a)

(b)

Fig. 4. Exemplar matching results for different near-duplication types. (a) Logo insertion and tone mapping. (b) Subtitle/logo insertion and resolution degradation. In each sub-figure, the top row is a reference video, the middle row shows initial matching results, and the bottom row shows final matching results. White rectangles are unmatched frames, and frames with dotted boundaries are mismatched ones.

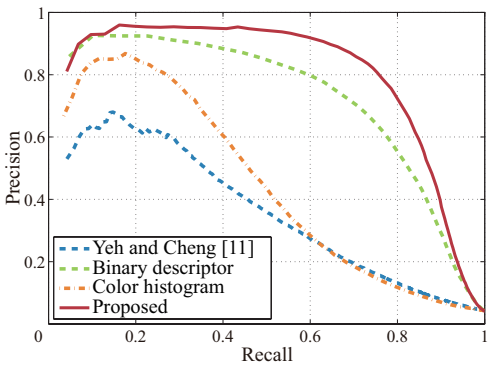


Fig. 5. Precision-recall curves with threshold adjustments.

all the descriptors, the same border removal scheme is applied as a preprocessing step.

Fig. 5 compares the precision-recall curves. The proposed descriptor outperforms the conventional descriptors significantly. While the conventional descriptors are vulnerable to near-duplication processes, the proposed descriptor detects near-duplicate frames reliably and suppresses the false detection of irrelevant frames.

4.2. Frame Matching Performance

We use 500 pairs of near-duplicate videos to assess the frame matching performance. We compute the precision and recall rates of matching results, by comparing them with manually obtained ground-truth correspondences. Also, we quantify the computational complexity, by averaging the execution times for matching the 500 pairs of videos.

The proposed frame matching algorithm is compared with the dynamic programming technique [9], which yields the globally optimal solution. Table 1 compares the precision rates, the recall rates, and the computational complexities. The proposed algorithm provides comparable precision and recall rates to the dynamic programming, while demanding significantly less computations.

Table 1. Comparison of frame matching performances

	Precision	Recall	Time (s.)
Dynamic programming [9]	0.96	0.97	0.102
Proposed algorithm	0.99	0.94	0.028

Fig. 4 shows examples of matching results of the proposed algorithm. The initial matching results are not accurate and do not include all corresponding frames due to logo and subtitle insertion, tone variation, and resolution degradation. However, after the iterative refinement, the proposed algorithm provides precise and dense matching results.

5. CONCLUSIONS

We proposed the ternary frame descriptor for near-duplicate video matching. Specifically, we partitioned a frame into patches and described the order relations of patch intensities with ternary digits, which were converted into bits for fast similarity measurement using the Hamming distance. Moreover, we developed an efficient and precise frame-level matching algorithm. We first formulated a cost function for the matching, composed of matching costs and adaptive unmatching costs. Then, we roughly determined initial matchings and refined those matchings iteratively to reduce the cost function monotonically. Experimental results demonstrated that the proposed ternary descriptor performs better than the conventional descriptors [11] and that the proposed matching algorithm achieves competitive performance with the global optimization technique [9], while demanding a significantly lower computational complexity.

6. REFERENCES

[1] YouTube. [Online]. Available: http://www.youtube.com/.

[2] Vimeo. [Online]. Available: https://vimeo.com/.

[3] S. Hu, "Efficient video retrieval by locality sensitive hashing," in *Proc. IEEE ICASSP*, 2005, vol. 2, pp. 449–452.

[4] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *Int. J. Comput. Vis.*, vol. 87, no. 3, pp. 316–336, 2010.

[5] L. Chen and F. W. M. Stentiford, "Video sequence matching based on temporal ordinal measurement," *Pattern Recog. Lett.*, vol. 29, no. 13, pp. 1824–1831, Oct. 2008.

[6] H.-K. Tan, C.-W. Ngo, R. Hong, and T.-S. Chua, "Scalable detection of partial near-duplicate videos by visual-temporal consistency," in *Proc. ACM Multimedia*, Oct. 2009, pp. 145–154.

[7] T. Chen, S. Jiang, L. Chu, and Q. Huang, "Detection and location of near-duplicate video sub-clips by finding dense subgraphs," in *Proc. ACM Multimedia*, Nov. 2011, pp. 1173–1176.

[8] C.-Y. Chiu, T.-H. Tsai, Y.-C. Liou, G.-W. Han, and H.-S. Chang, "Near-duplicate subsequence matching between the continuous stream and large video dataset," *IEEE Trans. Multimedia*, vol. 16, no. 7, pp. 1952–1962, Nov. 2014.

[9] Y.-Y. Lee, C.-S. Kim, and S.-U. Lee, "Video frame-matching algorithm using dynamic programming," *J. Electron. Imaging*, vol. 18, no. 1, pp. 1–3, Mar. 2009.

[10] M.-C. Yeh and K.-T. Cheng, "Video copy detection by fast sequence matching," in *Proc. ACM CIVR*, July 2009, pp. 45:1–45:7.

[11] M.-C. Yeh and K.-T. Cheng, "A compact, effective descriptor for video copy detection," in *Proc. ACM Multimedia*, Oct. 2009, pp. 633–636.

[12] M. Bertini, A. D. Bimbo, and W. Nunziati, "Video clip matching using MPEG-7 descriptors and edit distance," in *Proc. ACM CIVR*, July 2006, vol. 4071, pp. 133–142.

[13] A. Joly, O. Buisson, and C. Frelicot, "Content-based copy retrieval using distortion-based probabilistic similarity search," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 293–306, Feb. 2007.

[14] E. Maani, S. A. Tsaftaris, and A. K. Katsaggelos, "Local feature extraction for video copy detection in a database," in *Proc. IEEE ICIP*, Oct. 2008, pp. 1716–1719.

[15] S. Poullot, M. Crucianu, and O. Buisson, "Scalable mining of large video databases using copy detection," in *Proc. ACM Multimedia*, Oct. 2008, pp. 61–70.

[16] M. Douze, A. Gaidon, H. Jégou, M. Marszalek, and C. Schmid, "Inria-lears video copy detection system," in *Proc. TRECVID*, Nov. 2008.

[17] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, 2006.

[18] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proc. ACM Multimedia*, Oct. 2010, pp. 531–540.

[19] X. Tan and B. Triggs, "Enhanced local texture feature sets for face recognition under difficult lighting conditions," *IEEE Trans. Image Process.*, vol. 19, no. 6, pp. 1635–1650, June 2010.

[20] M. Nixon and A. Aguado, *Feature Extraction & Image Processing for Computer Vision*, Academic Press, 2012.

[21] Soku. [Online]. Available: http://www.soku.com/.