A Mix-domain Multimedia Algorithm in Video Segmentation

Yihan Sun
2010011356
CS&T05
syhlalala@gmail.com
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ABSTRACT

In this project, the general idea of a simplified cross-media algorithm which processes video segmentation is suggested. The aim of this algorithm is to detect shot changes in a video and cut it into segments, each of which leads to an independent scene or story.

Different from the baseline which only deals directly accessible feature, our algorithm introduced some high-level feature based on interest point matching under a projection transform, which involves skills in image processing. Our algorithm also takes both video and audio into consideration. A bi-classifier is applied to determine whether or not to make segmentations at a certain frame which finally contributes to the satisfactory performances in cartoons, sitcoms, teleplay and American dramas. Necessary analysis on the effects of different features is presented as the evidence proving the acceptable complements of video and audio.

Keywords

Cross-media, Video Segmentation

1. INTRODUCTION

In large-scale video analysis, it has become a trend to edit a given video into segments based on lexical information for the convenience of further uses like polishing or analyzing. However, a crucial challenge is that in most cases, a video consists of mix-domain information such as visual, auditory, as well as textual channels (subtitle and comments). All of these channels illustrate the features of that video in different perspectives, aiming at providing more information than any single one of them, which gradually makes the informational model much more complex.

In this project, a simple but efficient algorithm which produces a relatively good performance will be proposed. In order to simplify the problem, only visual and auditory signal are taken into consideration. The algorithm has two levels. A baseline based on directly accessible feature is first proposed. Improvements will be made by adding some high-level feature based on interest point matching under a projection transform into discussion. Improvement will be achieved by the high-level feature. In addition, a bi-classier is used for training the mix-domain features, which will determine whether or not to make a segmentation at a certain frame.

The general framework will be stated in section 2; the detailed algorithm will be elaborated in section 3; and brief experiments will be given in section 4. Section 6 presented some future work to do of this project. Section 7 concluded the project.

2. FRAMEWORK

2.1 Definitions

Assume a given video file is formed by some shots. A particular sequence of complete shots which tells a consistent event forms an event segment and lexically related event segments finally form the video.

To simplify the problem, we assume that whether to do a segmentation around t_0 is only associated with the frames in the neighborhood of t_0. We use $v(t)$ to represent the information in the video at time t.

For a video file $v(t)$ and a given time t_0, a sampling window $w(t) : R \to 2^R$ can be set around t_0 in order to get the video information of the period, namely, an interval of $[start(t_0), end(t_0)]$, where $start(t)$ denotes the earliest frame which can affect t, and likewise $end(t)$ denotes the latest one. An example for the sampling window is shown in Figure 1, in which $start(t_0) = t_0 - 0.5$ and $end(t_0) = t_0 + 0.5$. Thus here $w(t)$ represents a series of t-centered frames which last a second. Our aim is to extract feature from the initial video file $v(t)$ at a specified time t_0. We use a vector $\mathbf{V(t_0)}$ to represent the feature of the video at time t_0, where $\mathbf{V(t_0)}$ is extracted from the set $\{v(t)|t \in w(t_0)\}$. A decision function $f(\mathbf{V(t_0)})$ with a function domain of $\{0, 1\}$ should be trained to decide whether to make a segmentation around t_0.

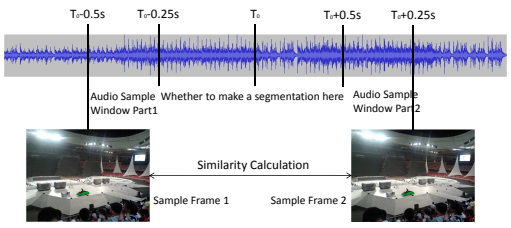

Figure 1: Sampling windows

2.2 Algorithm overview

As stated above, the baseline baseline based on directly accessible feature is first proposed. The features used in this part will be introduced in section 3.3.1. Then a high-level feature of corner correspondences will be introduced into the model. And in section 5, the algorithm advocated in

the project will include the discussion on auditory signal for further analysis.

A sketch map for the algorithm is shown in Figure 2.

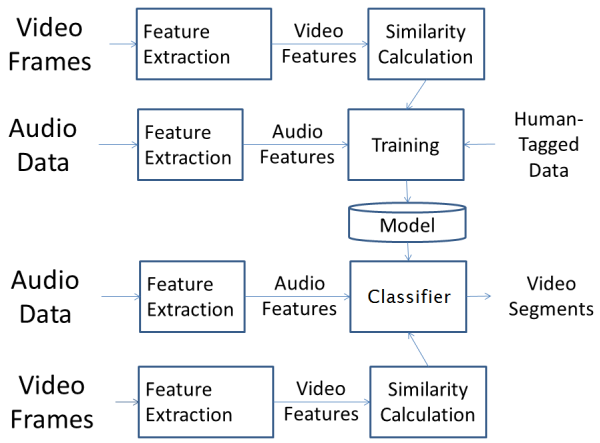

Figure 2: Sketch map for the algorithm

3. DETAILED ALGORITHM

In explaining the algorithm in detail, a classifier with directly accessible features will be stated first and the algorithm added high-level feature based on interest point matching under a projection transform will be placed after.

3.1 Directly Accessible Feature Extraction

To implement a simple classifier to detect shot change or event change, mix-domain feature must all be taken into consideration.

3.1.1 Visual Feature

First, color information should be used, and I just use the sum of difference in red, green and blue channel respectively as three features.

Moreover, I convert each frame into gray scale, and calculate the distance of the two neighboring frames. Two features will be extracted in this process. I previously get the difference at all the pixels between two frames, noted as matrix $\mathbf{D}$. The first feature is the sum of the absolute value of $\mathbf{D}$, represented as dis_{sum}. The second one is the number of nonzero value in $\mathbf{D}$, represented as dis_{nz}.

In summary, 5 features are used in this part.

3.1.2 Audio Feature

Pitch (extracted with *STRAIGHT* [4]), average amplitude and short-time power are used in the algorithm to represent an audio. Thus, 6 dimensions are in feature vector, namely, the pitch, average amplitude and power of the former and the later window respectively.

3.2 High-level Feature

3.2.1 Corner Point Detection

In the algorithm, interest points are treated as the feature of each frame. *Harris* [3] is first implemented as the corner detection algorithm. Interest points are chosen with the highest corner strength. Generally, vertices of a image fall into three categories:

Figure 3: The right images show interest points with the 100 highest corner strength, while the left one shows the same number of interest points selected with ANMS. The interest points in the left image have a more even distribution.

- **flat region**: They have no change in all directions.
- **edge**: They have no change along the edge direction yet significant change in other directions.
- **corner**: They have significant change in all direction.

In *Harris*, we get the strength of a point to become a corner. Without loss of generality, we will assume a grayscale 2-dimensional image is used. Let this image be given by $\mathbf{I}$. First, the partial derivatives of both directions should be calculated.

$$\mathbf{M} = \begin{pmatrix} I_x^2(i,j) & I_{xy}(i,j) \\ I_{xy}(i,j) & I_y^2(i,j) \end{pmatrix} \quad (1)$$

Then for the vertex (x, y), we can calculate the following function.

$$R(i,j) = det(\mathbf{M}) - k \times (trace(\mathbf{M}))^2; \quad (2)$$

According to $R(i, j)$, we can measure the probability of a vertex (i, j) to become a corner.

However, the distribution of the interest points shows a high density in some area (see the right part of Figure 3), which may probably result in information redundancy. An algorithm called adaptive non-maximal suppression [1] (ANMS) can be used to select a fixed number of interest points from each image. Only those containing a maximum of corner strength in a neighborhood will be chosen or kept.

The left part of Figure 3 gives the result of ANMS. Notice the distribution of the features different from the right part.

3.2.2 Partitioning Metric

Giving a metric to the partitioning between features and a corresponding transformation, some consensus strategy needs to be applied. Here *RANSAC* [2] is used.

Notice when the shot is slowly shifted, the two frames will correspond with the projective transformation. Then we transform all the corners of the first frame under this projective transformation and find how many of them match their image under the original mapping. The more the corners matches, the more the two frames are alike. Thus we would like to find the best projection transform.

First, the interest points are matched in the two neighboring frames into pairs with the method proposed in [5].

Denote Map as a mapping from the corners in picture 1 to those in picture 2. Then 4 pairs of corners are picked up randomly in the neighboring frames, and the homography is calculated, so that the projective transformation between them can be easily acquired. Next, all the corners of the first frame under this projective transformation need to be transformed and how many of them match their image under the original mapping need to be found out. The similarity of the two frames is indicated by the level of the corners' matching. A more corners' matching refers to a more similarity shared by the two frames. The number of matched point pairs is denoted as *best*. At the same time, the ratios of the transformed opposite side need to be calculated as another two indicators to demonstrate the distortion of the homography. A good map will made the ration to be near 1(if it is greater than 1, then get its reciprocal, and this ratio will be in the range[0,1]). The two ratios of the two pairs of opposite side need to be denoted as $indicator_1$ and $indicator_2$. They represent the distortions of the homography. With the same *best* value, greater $indicate_1$ and $indicate_2$ is preferred. The pseudo code is shown in Table 1.

Algorithm 1: RANSAC(picture1, picture2)

```
best = 0
bestInd = 0
for i=1:10000
    get 4 corners in picture1 randomly
    according to the 4 corners in picture 1 and their
    images in Map in picture 2, calculate the
    projective matrix H
    tot = 0
    for corners in picture1
        b ← Map(a)
        transform c under H as a
        distance = |a − b|
        if distance < η then tot++
    end for
    get indicator1 and indicator2 by H
    if tot > best then
        if indicator1 + indicator2 > bestInd
            best = tot
            bestH = H
            bestInd = indicator1 + indicator2
        end if
    end if
end for
```

Table 1: Algorithm 3: RANSAC

Specially, the value of *best* will illustrate the similarity of the two given pictures. In our test, 100 interest points are used for this algorithm and a simple example of successful matching (with $best = 47$) and failed matching (with $best = 8$) are shown in Fig. 4. All in all, 3 features are used in this part, namely, *best*, $indicator_1$ and $indicator_2$.

3.3 Classifier modeling

As whether to do segmentation is a binary decision variable, using bi-classifier is a sound approach with large manual-labeled training data. After extracting the feature of all the gaps between two neighboring windows as the input of classifier in our experiment, we decide whether to perform segmentation.

Figure 4: Examples of successful matching (the upper figure, 47/50 matched) and failed matching(the lower one, 8/50 matched).

3.3.1 Features

As introduced in 3.1 and 3.2, there are in all 15 features. The instruction of each feature is shown in Table 2.

I divided them into groups to study the influence of different types of features.

Group	No.	Name	Type
1:corner	1	$best$	visual
	2	$indicator_1$	visual
	3	$indicator_2$	visual
2:audio	4	$pitch_1$	audio
	5	$energe_1$	audio
	6	$amplitude_1$	audio
	7	$pitch_2$	audio
	8	$energe_2$	audio
	9	$amplitude_2$	audio
3:color	10	dif_{red}	visual
	11	dif_{green}	visual
	12	dif_{blue}	visual
4:location	13	dis_{sum}	visual
	14	dis_{nz}	visual

Table 2: Features in model

3.3.2 Classifier Implement

In this project, 3 kinds of classifiers are used, namely, decision tree, and KNN. A tool called sklearn in python is used. It also provide a package called feature selection to find best features in training data. 10-fold cross-validation is used.

4. EXPERIMENT

4.1 Data Set

The data is selected videos from cartoons, sitcoms, teleplay as well as American dramas. Each kind of data lasts about 40-60 minutes. The size of data sets can be seen in Table 3.

4.2 Baseline

Category	Data size
Cartoon	2147
sitcom	2448
teleplay	2405
American drama	3808

Table 3: The sizes of our data set.

In the baseline chosen for this project, only the directly accessible features are taken into consideration. For the four kinds of video files, the result is shown in Table 4.

category	method	precision	recall	F1
cartoon	svm	0.594(0.171)	0.493(0.293)	0.413(0.090)
	DT	0.459(0.105)	0.458(0.096)	0.454(0.083)
	KNN	0.654(0.155)	0.227(0.097)	0.332(0.122)
sitcom	svm	0.457(0.096)	0.547(0.341)	0.410(0.201)
	DT	0.650(0.026)	0.651(0.069)	0.649(0.041)
	KNN	0.654(0.054)	0.349(0.038)	0.453(0.035)
teleplay	svm	0.776(0.197)	0.559(0.298)	0.706(0.197)
	DT	0.738(0.054)	0.725(0.071)	0.728(0.037)
	KNN	0.703(0.047)	0.690(0.056)	0.694(0.033)
American drama	svm	0.612(0.046)	0.678(0.216)	0.630(0.114)
	DT	0.685(0.030)	0.669(0.038)	0.676(0.023)
	KNN	0.685(0.032)	0.644(0.044)	0.663(0.026)

Table 4: Precision and Recall for event detection of the baseline algorithm, in the brackets are standard deviations.

Figure 5 uses F1 value to illustrate the performance of different classifiers. Decision tree has the best performance, and in Table 4 we can also get that SVM has a unstable performance.

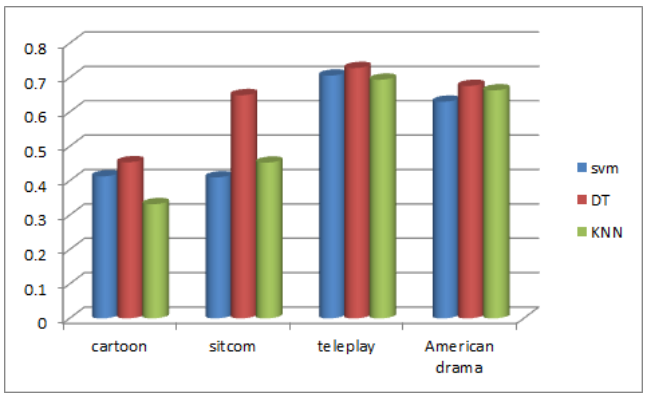

Figure 5: performance of baseline algorithm

4.3 With High-level Feature

Introducing the feature of *best*, $indicator_1$ and $indicator_2$, we can get the performance as in Table 5.

Obvious improvement is achieved by adding the three high-level feature in decision tree and SVM. No improvement is achieved by KNN. Use decision tree as an example, improvement is shown in Figure 6. Also, new features make the performance much more stable.

category	classifier	precision	recall	F1
cartoon	svm	0.482(0.144)	0.614(0.244)	0.422(0.119)
	DT	0.471(0.080)	0.450(0.050)	0.458(0.055)
	KNN	0.654(0.155)	0.227(0.097)	0.332(0.122)
sitcom	svm	0.552(0.191)	0.449(0.292)	0.490(0.119)
	DT	0.675(0.026)	0.677(0.042)	0.675(0.025)
	KNN	0.654(0.054)	0.349(0.038)	0.453(0.035)
teleplay	svm	0.823(0.079)	0.586(0.256)	0.675(0.233)
	DT	0.812(0.031)	0.786(0.043)	0.798(0.032)
	KNN	0.703(0.047)	0.690(0.056)	0.694(0.033)
American drama	svm	0.640(0.091)	0.472(0.301)	0.576(0.255)
	DT	0.700(0.031)	0.707(0.036)	0.703(0.031)
	KNN	0.685(0.032)	0.644(0.044)	0.663(0.026)

Table 5: Precision and Recall for event detection of the algorithm, in the brackets are standard deviations.

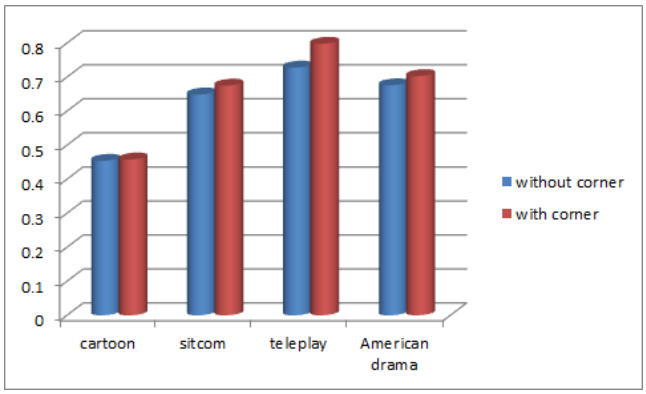

Figure 6: Improvement of decision tree

For decision tree has the best performance and is also very stable, in the following parts, I will focus on decision tree.

5. ANALYSIS

5.1 Influence of High-level Feature

Since the event is always made of a lot of shots, another experiment is designed. The test data of a sitcom is used and manually labeled some shot switches. Then we find the distribution of *best* at normal points, shot change points and event change points.

As can be seen from this experiment, only with *best* can shot switches be detected very well. With the stats in Figure 7, the performance is more abstractly explained. This figure illustrate that a high-level feature can strongly improve the performance of a classifier.

Another fact should also be noticed that *best* has the same distribution at shot change points and event shift points, which illustrate that with only *best*, no good performance in event detection will be gained. Event detection is a far more complex issue than shot detection.

5.2 Influence of Different Features

Some experiments were taken to compare the impact of different features. For decision tree is always more stable than the other two classifiers, here we only use decision tree to illustrate this topic. Taking auditory features into con-

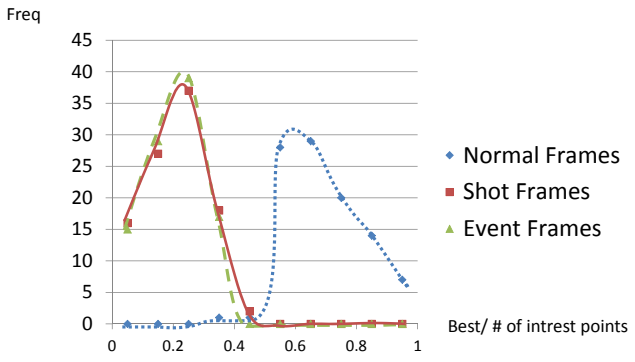

Figure 7: Explantation to baseline experiments

sideration, another experiment using **only** auditory features as the input of SVM is taken, which shows quite different performance to that of previous groups.

The results is given in Table 6. Comparing to the results in Table 5, neither video nor audio features separately show good performance in video segmentation, but by combining these features together, a better performance can be achieved, showing that video and audio features well complement each other.

For the result shown in section 5.1, here I only use *best* as the corner feature in shot change detection.

feature	cartoon	sitcom	teleplay	American drama
all	0.458	0.675	0.798	0.703
corner	0.064	0.634	0.712	0.652
audio	0.34	0.361	0.272	0.34
color	0.387	0.629	0.689	0.623
location	0.391	0.516	0.714	0.629

Table 6: different F1 gained by different groups of feature

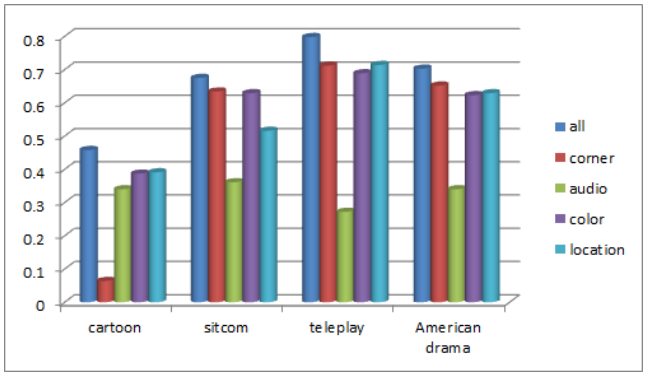

Figure 8: different F1 gained by different groups of feature

Figure 8 illustrate the result shown in Table 6. From 6, we can get that corner information is always the group to have best performance and overwhelms the others, which indicate the importance of the high-level feature performance. However, in cartoons dataset, it has the worst performance, and F1 value is even less than 0.1. That is because in cartoons, corner points are harder to detect. Cartoons are made by drawing, thus no complex high frequency information can be used to detect and match corners. As a result, Corner information fails to work well. However, in real-world dataset, the feature and neighborhood of a corner is very specified, which make the corner information overwhelm other groups.

Also, by feature selection package in python sklearn, I get the ranking of the importance of each features, which will be shown in Table 7.

ranking	cartoon	sitcom	teleplay	American drama
1	dif_{green}	*best*	*best*	*best*
2	dif_{red}	dif_{red}	dis_{sum}	dis_{sum}
3	dif_{blue}	dif_{blue}	dif_{red}	$indicator_2$
4	dis_{sum}	dif_{green}	dif_{blue}	dif_{blue}
5	*best*	$indicator_2$	$indicator_2$	dif_{green}
6	$indicator_2$	dis_{sum}	dif_{green}	dif_{red}
7	$indicator_1$	$indicator_1$	$indicator_1$	$indicator_1$
8	dis_{nz}	dis_{nz}	dis_{nz}	dis_{nz}
9	$pitch_1$	$pitch_2$	$amplitude_2$	$energy_2$
10	$amplitude_1$	$amplitude_2$	$energy_2$	$amplitude_2$
11	$energy_1$	$pitch_1$	$pitch_1$	$amplitude_1$
12	$pitch_2$	$energy_2$	$amplitude_1$	$pitch_2$
13	$amplitude_2$	$energy_1$	$energy_1$	$energy_1$
14	$energy_2$	$amplitude_1$	$pitch_2$	$pitch_1$

Table 7: Ranking of different features.

Table 7 indicates that corner information and color information is always more important than others, especially in real world videos. It is much better than other directly accessible features. Color is important, too. Also, when judging a shot change by color, different colors has different affects. Red always has a larger influence in real world video than the other two colors, in the next place is blue. Green is the last one.

Audio information is never more important than any of the other features in shot detection. This is obvious. When people judge shot changes, they depend more on visual information, less on auditory one. Another important reason is that all the auditory features are very low-level, which cannot well represent the feature of auditory information, and also, pitch extraction is not very precise.

Does this mean auditory information is useless in our model? The answer is no. Table 8 and Figure 9 show that without auditory information, the performance has a slight decrease.

	cartoon	sitcom	teleplay	American drama
all feature	0.458	0.675	0.798	0.703
without audio	0.432	0.656	0.788	0.691

Table 8: F1 with or without audio feature

5.3 Further Explain

Obviously visual information determines the shot change detection, however, why color and location are the suitable

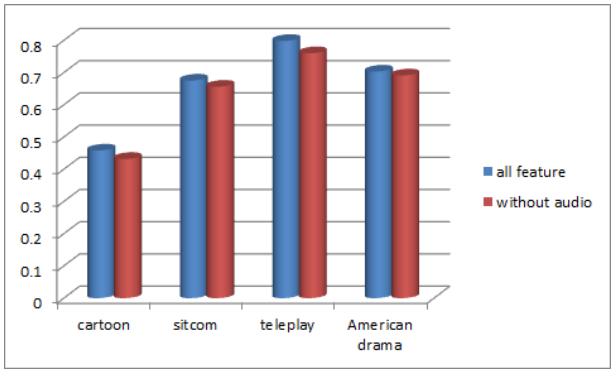

Figure 9: F1 with or without audio feature

Figure 10: How corner points work in camera shifting situation. From up to down is translation, rotation and scaling.

feature? The explain can be as follows.

First, imagine the situation that no shot change, thus some similarity should be found between the two windows. If we can properly define the similarities and use them as features to train the classifier, satisfactory result will be achieved. Empirically, there are mainly two situations.

The first situation is no shot shift but roles moving. Two features can help to find the similarity. One is the corner points of the background, and the second is the color distribution. Always, the corner point of the background may not have very high strength, thus hard to detect. In these situations, color is the main metric to judge the similarity.

Another situation is the camera moves around the roles, thus making the background change, but the figures in the scene have almost no movements. The corner points play a role in this situation. Translation, rotation, and scaling are the most common change. They all fit projection transformation. Figure 10 shows the result in translation, rotation, and scaling respectively.

6. FUTURE WORK

The features extracted in this project is far from enough. First, more futures can be gained. For example, if we map the RGB space into HSV space, hue, saturation, and lightness can all be specified, which may contribute to our algorithm. Also, we can cut the image into blocks to get color feature, which will represent the location of colors.

Also, the classifier is too general. A feasible solution is to find a suitable kernel in SVM. That requires more work and study on the features.

Also, shot change detect is far from scene change or event change detection. Higher level detection in scene, or even sematic detection is a challenge.

In the algorithm, lexical information in the video and prior information about segment length distribution are not considered in a high enough level, and these features may be included in the future works.

7. CONCLUSION

A new algorithm of processing cross-media video segmentation is advocated in this project. An algorithm based on direct accessible feature extracting and matching as well as threshold detection is used as the baseline of the algorithm. With high-level features and a bi-classier handling the cross-media information, the cross-media algorithm suggested tends to be sound and solid.

8. REFERENCES

[1] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 510–517. IEEE, 2005.

[2] M. A. Fischler and R. C. Bolles. 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.

[3] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[4] H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno. Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 3933–3936. IEEE, 2008.

[5] D. P. Mcreynolds and D. G. Lowe. Rigidity checking of 3d point correspondences under perspective projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:1174–1185, 1996.