LaTeX Author Guidelines for CVPR Proceedings

Mouse Lee
jl27@princeton.edu

Jonathan Metzman
jmetzman@princeton.edu

Bo Moon
bhmoon@princeton.edu

Abstract

The aim of this project is to develop a program for visual speech recognition by identifying spoken words based on lip motions. These lip motions are tracked using an external library for Constrained Local Models. The lip motion of a user is extracted using this library, then compared against a precompiled database consisting of lip motions corresponding to certain words. We present algorithms for speech detection and calculating an error metric between two different lip motions, as well as discussing various obstacles and assumptions present in handling visual data, such as unwanted geometric transformations of the face. The performance of this program is tested using a limited dictionary and several users.

1. Introduction

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sticky tape to attach your artwork to the paper), so all authors should read this new version.

1.1. Language

All manuscripts must be in English.

1.2. Dual submission

Please refer to the author guidelines on the CVPR 2015 web page for a discussion of the policy on dual submissions.

1.3. Paper length

For CVPR 2015, the rules about paper length have changed, so please read this section carefully. Papers, excluding the references section, must be no longer than eight pages in length. The references section will not be included in the page count, and there is no limit on the length of the references section. For example, a paper of eight pages with two pages of references would have a total length of 10 pages. **Unlike previous years, there will be no extra page charges for CVPR 2015.**

Overlength papers will simply not be reviewed. This includes papers where the margins and formatting are deemed to have been significantly altered from those laid down by this style guide. Note that this LaTeX guide already sets figure captions and references in a smaller font. The reason such papers will not be reviewed is that there is no provision for supervised revisions of manuscripts. The reviewing process cannot determine the suitability of the paper for presentation in eight pages if it is reviewed in eleven.

1.4. The ruler

The LaTeX style defines a printed ruler which should be present in the version submitted for review. The ruler is provided in order that reviewers may comment on particular lines in the paper without circumlocution. If you are preparing a document using a non-LaTeX document preparation system, please arrange for an equivalent ruler to appear on the final output pages. The presence or absence of the ruler should not change the appearance of any other content on the page. The camera ready copy should not contain a ruler. (LaTeX users may uncomment the `\cvprfinalcopy` command in the document preamble.) Reviewers: note that the ruler measurements do not align well with lines in the paper — this turns out to be very difficult to do well when the paper contains many figures and equations, and, when done, looks ugly. Just use fractional references (e.g. this line is 095.5), although in most cases one would expect that the approximate location will be adequate.

1.5. Mathematics

Please number all of your sections and displayed equations. It is important for readers to be able to refer to any particular equation. Just because you didn't refer to it in the text doesn't mean some future reader might not need to refer to it. It is cumbersome to have to use circumlocutions like "the equation second from the top of page 3 column 1". (Note that the ruler will not be present in the final copy, so is not an alternative to equation numbers). All authors will benefit from reading Mermin's description of how to write mathematics: `http://www.pamitc.org/documents/mermin.pdf`.

1.6. Blind review

Many authors misunderstand the concept of anonymizing for blind review. Blind review does not mean that one must remove citations to one's own work—in fact it is often impossible to review a paper unless the previous citations are known and available.

Blind review means that you do not use the words "my" or "our" when citing previous work. That is all. (But see below for techreports.)

Saying "this builds on the work of Lucy Smith [1]" does not say that you are Lucy Smith; it says that you are building on her work. If you are Smith and Jones, do not say "as we show in [7]", say "as Smith and Jones show in [7]" and at the end of the paper, include reference 7 as you would any other cited work.

An example of a bad paper just asking to be rejected:

> An analysis of the frobnicatable foo filter.
>
> In this paper we present a performance analysis of our previous paper [1], and show it to be inferior to all previously known methods. Why the previous paper was accepted without this analysis is beyond me.
>
> [1] Removed for blind review

An example of an acceptable paper:

> An analysis of the frobnicatable foo filter.
>
> In this paper we present a performance analysis of the paper of Smith *et al.* [1], and show it to be inferior to all previously known methods. Why the previous paper was accepted without this analysis is beyond me.
>
> [1] Smith, L and Jones, C. "The frobnicatable foo filter, a fundamental contribution to human knowledge". Nature 381(12), 1-213.

If you are making a submission to another conference at the same time, which covers similar or overlapping material, you may need to refer to that submission in order to explain the differences, just as you would if you had previously published related work. In such cases, include the anonymized parallel submission [2,] as additional material and cite it as

> [1] Authors. "The frobnicatable foo filter", F&G 2014 Submission ID 324, Supplied as additional material `fg324.pdf`.

Finally, you may feel you need to tell the reader that more details can be found elsewhere, and refer them to a technical report. For conference submissions, the paper must stand on its own, and not *require* the reviewer to go to a techreport for further details. Thus, you may say in the body of the paper "further details may be found in [2]". Then submit the techreport as additional material. Again, you may not assume the reviewers will read this material.

Sometimes your paper is about a problem which you tested using a tool which is widely known to be restricted to a single institution. For example, let's say it's 1969, you have solved a key problem on the Apollo lander, and you believe that the CVPR70 audience would like to hear about your solution. The work is a development of your celebrated 1968 paper entitled "Zero-g frobnication: How being the only people in the world with access to the Apollo lander source code makes us a wow at parties", by Zeus *et al.*

You can handle this paper like any other. Don't write "We show how to improve our previous work [Anonymous, 1968]. This time we tested the algorithm on a lunar lander [name of lander removed for blind review]". That would be silly, and would immediately identify the authors. Instead write the following:

> We describe a system for zero-g frobnication. This system is new because it handles the following cases: A, B. Previous systems [Zeus et al. 1968] didn't handle case B properly. Ours handles it by including a foo term in the bar integral.
>
> ...
>
> The proposed system was integrated with the Apollo lunar lander, and went all the way to the moon, don't you know. It displayed the following behaviours which show how well we solved cases A and B: ...

As you can see, the above text follows standard scientific convention, reads better than the first version, and does not explicitly name you as the authors. A reviewer might think it likely that the new paper was written by Zeus *et al.*, but cannot make any decision based on that guess. He or she would have to be sure that no other authors could have been contracted to solve problem B.

FAQ: Are acknowledgements OK? No. Leave them for the final copy.

1.7. Miscellaneous

Compare the following:

`$conf_a$`	$conf_a$
`$\mathit{conf}_a$`	conf_a

See The TEXbook, p165.

The space after *e.g.*, meaning "for example", should not be a sentence-ending space. So *e.g.* is correct, *e.g.* is not. The provided `\eg` macro takes care of this.

When citing a multi-author paper, you may save space by using "et alia", shortened to "*et al.*" (not "*et. al.*" as "*et*" is a complete word.) However, use it only when there

Figure 1. Example of caption. It is set in Roman so that mathematics (always set in Roman: $B \sin A = A \sin B$) may be included without an ugly clash.

are three or more authors. Thus, the following is correct: “ Frobnication has been trendy lately. It was introduced by Alpher [2], and subsequently developed by Alpher and Fotheringham-Smythe [2], and Alpher *et al*. [2].”

This is incorrect: “... subsequently developed by Alpher *et al*. [2] ...” because reference [2] has just two authors. If you use the `\etal` macro provided, then you need not worry about double periods when used at the end of a sentence as in Alpher *et al*.

For this citation style, keep multiple citations in numerical (not chronological) order, so prefer [2] to [2].

2. Motivation

2.1. Why do this

We found VSR to be a worthy problem for a couple of reasons. A VSR system can be used to recognize speech in scenarios where only visual data is kept, such as in Closed Circuit TV recordings. A more common use case than recognizing speech in video recording without audio is recognizing speech in a recording with a large amount of background noise. Background noise makes audio based speech recognition considerably more difficult, so a vision or hybrid speech recognition system could be more accurate in noisy scenarios [2].

2.2. What makes this difficult

We tried to build a system that can determine which words, from a predefined set of words, or dictionary, are being said. This kind of system could allow computer users to perform simple tasks, such as navigating to a specific website, by speaking, even in the presence of noise, such as while listening to music. Like most other computer vision problems, “VSR is Hard” . A VSR system must be able to determine the words that are being spoken based on the motion of a user’s mouth. This can actually be quite hard in the real world because of things like diferences between the lighting or camera in the testing and training data. Recognizing the speech of real people can be difficult becuase of behaviors like moving their face. In the real world there also things that make tracking the motion of the mouth more difficult such as makeup, glasses, facial hair, or teeth. We beleive that our system is resilient to these differences, and performs will in real conditions. One thing that is important for practical reasons that our system is capable of is recognizing the speech of a user without requiring training on data supplied by the user themself. !!WHAT ABOUT USER NOT FACING THE CAMERA!! There are other issues that seem to be insurmountable for our system such such as poor lighting and occlusion; we think a speech recognition system that also used audio data would be most robust in these scenarios.

3. Related Work

3.1. What has been done

Speech recognition has gone from academic research interest to an essential part of the way user interact with computing devices [3]. Most of these technologies rely solely on acoustic data. However, an obvious drawback of these systems is that their performace is severly degraded by noise [1]. Research has shown that systems that use visual data in place of, or in addittion to audio data, can significantly outperform systems that only use audio data in the presence of noise [3].

3.2. What are the problems

Most other research, with a few exceptions focuses mainly on acoustic recognition, with visual recognition used as a secondary means of recognition. We thought that using a purely vision based approach would be an interesting challenge, and provide some of the advantages outlined above. Another difference between our approach and those of most other VSR systems is that most of them try to detect visual phenomes, or visemes, rather than actual words. While this approach is more generalizable and would allow detection of words that the system has not been explicitly trained on, we suspected that this approach may be less accurate for recognizing specific words than the approach we decided to take. One study that used this method was able to acheive 72.73% recognition of vowels, since the percentage would drop for specific words we felt the usefulness of this approach is quite limited for many use cases [4]. A final advantage our implementation has over nearly any other implementation is portability. Since our implementation can run in web browsers it can run on virtually every platform in common use today.

Figure 2. Example of a short caption, which should be centered.

4. Algorithm

4.1. Overview

First we will present an outline of how the program operates, and then afterwards discuss major problems, assumptions, and algorithms. The program is outlined as follows:

1. Initialize dictionary: The user first must compile a database of known words and the associated lip motions so queries may be made later.

2. Begin video capture, initialize mouth threshold and nose bridge length: The program displays a live feed and also initializes two parameters. The program also uses an external library to track the users lips.

3. Detect when word has been spoken: The program detects when the user has spoken a word.

4. Capture all relevant frames: After the user has finished speaking, all frames associated with the word are stored to be used as a query against the database.

5. Compare query to each word in the database: A metric is computed to determine how similar the query is to a word in the database as measured by their lip motions.

6. Use modified k nearest neighbors to return the best matching word: The computed metrics are used to find a group of the most similar words, which is then used to find the best matching word.

4.2. Obtaining and Representing Data

Our program makes use of the Javascript clmtracker library (https://github.com/auduno/clmtrackr). It uses Constrained Local Models, a technique for finding statistical estimates of coordinates on an object based on its shape, combined with trained support vector machines to identify and track many facial landmarks, as seen below. INCLUDE IMAGE!!! The CLM tracker library is used to identify and track 18 points on the lip. These 18 points are represented by and pixel coordinates for each frame; thus, the program represents each frame as an array of 18 points, also interpreted as a matrix of 18 rows and 2 columns:

$$f = [(x_1, y_1), (x_2, y_2), \ldots, (x_{18}, y_{18}) \qquad (1)$$

The program represents a word as an array of all consecutive frames in which the user was speaking, plus 25 additional frames before and after speaking as a buffer to capture initial and final lip motions. For instance, if the program detected that the user spoke for frames, then the word is stored as

$$w = [f_1, f_2, \ldots, f_{n+49}, f_{n+50}] \qquad (2)$$

where the first 25 frames are immediately before when the program detects the start of speech, and the last 25 frames are immediately after when the program detects the end of speech. The value 25 was chosen since that is the average frame rate of the webcam used.

4.3. Geometric Transformations

The points obtained by the CLM tracker are stored in pixel coordinates. As such, these coordinates are highly sensitive to various geometric transformations such as rotations and translations, which make the points difficult to handle for algorithms and computations. Ideally, lip point coordinates will change only when there is lip motion, as in speech. We can implement some measures to reduce or ignore undesirable motion.

4.3.1 Translations

Translations occur when the user moves his/her face around the screen (assuming the face does not turn/rotate). INCLUDE IMAGE Consider the pixel coordinates of the right corner of the lip in the figure above. Even though the lip is in the same position on the face, the pixel coordinates before and after the translation are different because the entire face has moved. However, note that the position of that particular point has not changed relative to any other point on the face since all points on the face have been translated the same amount. As such, we can make lip point coordinates resistant to translations by transforming the coordinates into a new coordinate system with the origin somewhere on the face. This way, translating the face would translate the entire coordinate system, so no lip point coordinates would be changed. In our program we accomplish this by imposing a coordinate system on the tip of the nose, although any facial feature would have worked. Specifically, for every lip point (x, y) in pixel coordinates and the pixel coordinates of the tip of the nose $(nose_x, nose_y)$, the new coordinates are a position vector in the form

$$p = (nose_x - x, nose_y - y) \tag{3}$$

4.3.2 Scaling

INCLUDE IMAGE!!! The face in the video changes scale when the user changes his/her distance to the camera, e.g. the face becomes larger when the user is closer and shrinks when the user is farther away. The coordinates of are still sensitive to scaling. However, when the face changes scale, all lengths on the face are scaled by the same amount, so the face before and after scaling are geometrically similar. Therefore, by using simple geometry, we can find ratio of lengths on the face to calculate this scaling factor. The CLM tracker also tracks the nose bridge. During the calibration phase, the nose bridge length of the user face is recorded. Later, during the testing phase, the scaling factor is calculated by

$$s = \frac{\text{recorded nose length}}{\text{current nose length}} \tag{4}$$

where is the nose length recorded during calibration and is the nose length of the face in the current frame. Afterwards, this scaling factor is used to scale all lip points per frame as such:

$$p' = s \times p = (s(nose_x - X), s(nose_y - y)) \tag{5}$$

4.3.3 Rotations

Rotations occur when the user rotates his/her face. This is a significant problem since if the users face rotates too much, the coordinate system based on the nose would not rotate with the face, so coordinates of each point would be incorrect. The simplest solution to this problem is to avoid having rotations altogether. As such, we require that the users face looks straight on into the camera, i.e. the optical axis of the camera is orthogonal to the plane of the face, and the head is not tilted. Calculating p' is a way to make the coordinates less sensitive to changes so that we do not have to be concerned with geometric problems in later algorithms. However, in practice p' still changes slightly due to these transformations. For instance, when the user translates his/her face in the video, the user will naturally turn to face the camera the entire time. This rotation has been observed to cause the coordinates of p' to fluctuate by 4-6 units, which is not very significant. Nevertheless, the program performs best when the users face is in the center of the screen, looking straight at the camera without rotating, and as stationary as possible.

4.4. Calibration

Calibration is the procedure of compiling a dictionary of known words so that queries can be made later to find the best matching word. There are several problems of calibration that will be discussed later, such as how speech is detected and what particular words ought to be stored in the dictionary.

Calibration is performed before making any queries, but after calibration is done it does not need to be repeated. The program works best if the same user who calibrates the dictionary also makes the queries, so it is recommended that a new user to calibrate his/her own dictionary. During calibration, the user will prepare a list of words he/she wishes to be in the dictionary. The user then speaks each of these words multiple times to camera. The program detects the speech and records an array of frames and prompts the user to type what word was spoken. Afterwards, this array and string are stored as a pair in a database. In the end, if there are W words and each is said T times, the database will have WT entries where each entry is a tuple of an array of frames and a string, but there will only be W unique strings in the database. These duplicate strings exist for use in a modified k nearest neighbors algorithm, to be discussed later. As for implementation details, the database used in our program was ”Firebase” and we had 5 distinct words that were each repeated 6 times for a total of 30 entries in the dictionary.

4.5. Video Capture and Speech Detection

There are some basic miscellaneous assumptions associated with the capture and display of video: The user and camera should be in a well-lit but not too bright setting; the face should be visible (glasses and hair are acceptable but nothing covering the mouth); the user should not speak too quickly, and slowly enough so that the video shows the lip moving without skipping or blurring motions.

The program needs to determine when the user has started and finished talking so that all frames containing lip motions can be stored. First, to determine whether the mouth is open or closed, we use the distance between the centers of the top and bottom lips, which we refer to as the mouth height. Before speech detection begins, we require the user to have his/her mouth closed and record the square of the mouth height. During speech detection, for each frame we calculate the square of the current mouth height. If it is greater than the recorded value, then this means the mouth is open; otherwise, the mouth is closed. We use squared distances because this makes the program more sensitive to small changes in motion; for instance, if the camera observes the lips move by 2 pixels, then the program registers this motion as 4 units.

For speech detection, we use a combination of counting frames and threshold values. In short, speech is detected if this sequence of events is followed:

1. Mouth opens: Determined when the previous frame detects a closed mouth and current frame is open.

2. Mouth remains open for at least a certain (small) number of frames: If the mouth is not open for at least that many frames, most likely the program accidentally detected a false open mouth. Otherwise the mouth is still in motion right now.

3. Mouth remains closed for a certain (relatively large) number of frames: If the mouth is not closed for at least that many frames, most likely the mouth only momentarily closed to pronounce a plosive sound (e.g. baboon), so this does not count as the user being done speaking. Otherwise this indicates the mouth is done speaking.

An array is used to store the relevant frames in the meantime, and when the user has finished speaking, these frames are processed.

4.6. Comparing Words

When the query is obtained, it is stored as an array of frames q. Furthermore, all words in the dictionary database are stored as a tuple of a string representing the word itself and an array of frames spoken during calibration. In order to find the best matching word in the dictionary, we need some way to measure the similarity between the query array q and a dictionary word array w.

We can describe this problem geometrically, but to simplify the description, we will consider only one point on the lip, then later generalize to all 18 lip points. For now, let $q = (q_1, q_2, q_3, \ldots, q_n)$ be a sequence of coordinates such that $q_i = (x_i^q, y_i^q)$ are the coordinates of a lip point during the ith frame of speaking a query, and there are n such coordinates in total. Similarly, let $w = (w_1, w_2, w_3, \ldots, w_n)$ be a sequence of frames such that $w_i = (x_i^w, y_i^w)$ are the coordinates of a lip point during the ith frame of speaking a dictionary word, and there are m such coordinates in total. INCLUDE IMAGE!! If the coordinates stored in the array q are plotted and connected, they form the curve that the lip point travels across those frames; the same is true for w. As such, the problem of finding the similarity between a query and a word can be posed as determining how similar two curves are. If $n = m$ (that is, if the query and word are the exact same number of frames) then a natural metric to use is the sum of squared distances between the corresponding coordinates:

$$error(q, w) = \sum_{i=1}^{n} \|q_i - w_i\|^2 \tag{6}$$

where $||q_i - w_i|| = \sqrt{(x_i^q - x_i^w)^2 + (y_i^q - x_i^w)^2}$ is the distance function. With this, a smaller error corresponds to a more similar curve. However, this metric is problematic because it is exceedingly rare that a user will say two words for exactly the same number of frames, even if those words were the same. The inconsistency in the length of the arrays also makes it difficult to apply a machine learning algorithm, since a feature vector would have to be engineered to compensate for arrays of variable dimensions. As such, another metric is needed. INCLUDE PICTURE!! Consider the two curves q and w for $n < m$ (that is, the query is spoken for fewer frames than the word in the dictionary). Since q is shorter than w, it is possible to align q in various ways such that the relative ordering of the elements of q is preserved but the corresponding elements in w are different, as in the figure. For each of these alignments, we can still calculate the sum of squared distances by taking pairs of corresponding entries, despite the fact that q is shorter than w. As such, we define a similarity metric between q and w to be the minimum possible score obtainable among all possible alignments of q to w and taking the sum of squared distances. Interpreted geometrically, this metric is calculated by splitting the curve q into multiple strands, then trying to align each of those strands against w while maintaining the correct order of strands, as in Figure 6. Furthermore, we must attempt all possible splits and all possible number of strands that can be made from q. INCLUDE PICTURE!!! To calculate this metric, we use a dynamic programming algorithm. Let $M(i, j)$ be the similarity metric between the sequences $(q_1, q_2, q_3, ldots, q_i)$ and $(w_1, w_2, w_3, ldots, w_j)$. In other words, $M(i, j)$ is the smallest score obtainable by aligning the first i coordinates of q to the first j coordinates of w. We will try to align q to w starting from the end, then align recursively. For q_i, there are only two possible options when aligning it to w_j: either q_i is aligned with w_j, or it is not aligned. If q_i is aligned with w_j, these two coordinates form a pair that contributes an amount of $\|q_i - w_i\|$ to the final score furthermore, the problem is now reduced to aligning $(q_1, q_2, q_3, ldots, q_i)$ to $(w_1, w_2, w_3, ldots, w_j)$. If q_i is not aligned with w_j, then this means that q_i must be aligned with some coordinate of w that comes before w_j; this is described by the subproblem $(q_1, q_2, q_3, ldots, q_i)$ and $(w_1, w_2, w_3, ldots, w_j)$ Thus, we can represent these relationships as

$$M(i,j) = \min\{M(i-1,j-1) + \|q_i - w_j\|^2, M(i,j-1)\} \quad (7)$$

The base cases are occur when i ¿ j or i = j = 0. When i ¿ j, this means that there are too many coordinates in q to align against w, so it is impossible to make an alignment; thus, $M(i,j) = \infty$. When i = j = 0, this is the trivial case of aligning an empty sequence against an empty sequence, so $M(i,j) = M(0,0) = 0$. The time and space complexity of this algorithm is $O(nm)$ since this dynamic programming algorithm uses memoization to fill an n by m table (note this table would be a lower triangular matrix because of the base cases).

Using this similarity metric, we can compute the score between q and w to be $M(i,j)$ for when $n < m$ (we will discuss the case when $n \geq m$ shortly). This is only for the simplified case of one lip point, but we can generalize this to all 18 lip points. Let $q = (f_1^q, f_2^q, f_3^q, \ldots f_n^q)$ be a sequence of frames that constitutes the spoken query, where $f_i^q = [(x_1^q, y_1^q), (x_2^q, y_2^q), \ldots, (x_{18}^q, y_{18}^q)$ is the array of 18 lip points as defined earlier. Define w analogously. The similarity metric is now

$$M(i,j) = \min\{M(i-1,j-1) + \|f_i^q - f_j^w\|, M(i,j-1)\} \quad (8)$$

where we define

$$\|f_i^q - f_j^w\| = \sum_{1}^{18} (x_i^q - x_i^w)^2 + (y_i^q - y_j^w)^2 \quad (9)$$

or in words, the sum of squared distances between all 18 pairs of corresponding lip points in two frames.

Since $M(i,j)$ is the minimum sum of squared differences between curves, it is actually a geometric error metric between q and w. This error metric can be computed for the query compared against each word in the dictionary. As such, a word with a relatively small error metric means that it has a similar shaped path to the query.

It appears this algorithm works only for $n < m$, or when the query is spoken faster than the word being compared. One obvious solution would be to swap q and w and call $M(m,n)$ instead. However, this does not address the main weakness of this algorithm: if $n << m$, it is likely that $M(m,n)$ will be very low, so q will be calculated to be very similar to w. This is because there are many possible alignments if the query is spoken much faster than the word, so it is more likely for a smaller score to be found. As such, the real issue is making sure that the ratio of n to m is an appropriate amount.

References

[1] M. Cooke, P. Green, L. Josifovski, and A. Vizinho. Robust automatic speech recognition with missing and unreliable acoustic data. *Speech Communication*, 34(3):267 – 285, 2001.

[2] S. Dupont and J. Luettin. Audio-visual speech modeling for continuous speech recognition. *Journal of Foo*, 2014.

[3] P. Lison and R. Meena. Spoken dialogue systems: The new frontier in human-computer interaction. *XRDS*, 21(1):46–51, Oct. 2014.

[4] S. Werda, W. Mahdi, and A. B. Hamadou. Lip localization and viseme classification for visual speech recognition. *CoRR*, abs/1301.4558, 2013.