Computer Vision and Sign Language Research

AKLI Melissa
PASECHNIUK Dmitrii

April 18, 2024

UGA
Université
Grenoble Alpes

Contents

1 Introduction:

1.1 Computer Vision:

Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer. [24]

Computer vision is an exciting field of computer science that aims to give computers the ability to understand and interpret visual information, much like humans do. It finds applications in various domains, such as object recognition, motion detection, augmented reality, robotics, surveillance, medicine, and many others

1.2 History:

Computer vision has a rich history that traces back to the 1960s, when pioneering researchers began exploring the potential of computers to understand and interpret visual information. Despite the limited computing power and technological constraints of the time, these early visionaries laid the groundwork for the field by developing basic algorithms for tasks like edge detection and pattern recognition.

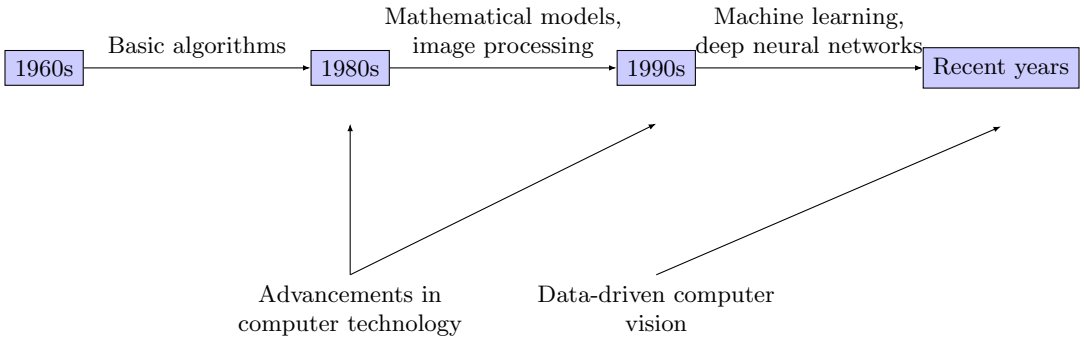


Figure 1: Evolution of Computer Vision

Over the decades, with rapid advancements in computer technology, computer vision has undergone significant evolution. Researchers have developed more sophisticated techniques for image processing and analysis, leveraging advances in image processing, pattern recognition, machine learning, and artificial intelligence.

In the 1980s and 1990s, the use of mathematical models and image processing techniques led to significant progress in areas such as image segmentation, stereo correspondence, and 3D reconstruction. These advancements opened up new possibilities in fields like virtual reality, 3D modeling, and geometry-based computer vision.

In recent years, the evolution of computer vision has been marked by the emergence of machine learning and deep neural networks. These new approaches, known as data-driven computer vision, have greatly improved the performance of computer vision systems in various domains. Deep neural networks, in particular, have achieved remarkable results in tasks such as image classification, object detection, and facial recognition .

In conclusion, computer vision has come a long way since its humble beginnings. Thanks to technological advancements and the rise of machine learning, this field continues to push

the boundaries of what is possible in understanding and interpreting visual information. Computer vision plays an increasingly important role in our daily lives and holds promising advancements for the future

1.3 Overview and application in other field:

Computer vision, as a field of artificial intelligence, has applications across numerous domains, demonstrating its versatility and wide-ranging utility. Its ability to analyze and interpret visual data has led to groundbreaking advancements in various sectors, including healthcare, autonomous vehicles, and agriculture.

Figure 2: autonomous vehicles source for Figure 2,

Figure 3: Agriculture source for figure 3,

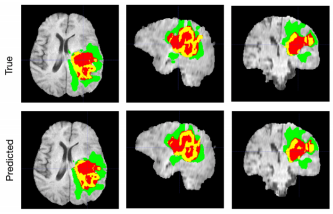


Figure 4: Healthcare source for figure 4

According to Tian in Computer Vision for Agriculture: A Review[25] computer vision technologies revolutionize farming practices by providing valuable insights into crop health, yield estimation, and resource management. By analyzing aerial and ground-based images, computer vision systems can detect plant diseases, assess crop growth and health, and optimize irrigation and fertilizer usage.This enables farmers to make data-driven decisions, improve crop productivity, and reduce environmental impact.

In the realm of autonomous vehicles, computer vision is instrumental in enabling perception systems for safe and efficient navigation. Through the analysis of sensor data, including cameras and LiDAR, computer vision algorithms can identify objects, detect obstacles, and recognize traffic signs and signals.As stated in Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art[15],these capabilities are crucial for the development of autonomous driving technologies, enhancing road safety and transportation efficiency.

In healthcare applications, computer vision plays a crucial role in aiding clinical procedures and diagnostics. For instance, it enables the detection and monitoring of diseases such as cancers, cardiovascular disorders, and neurological conditions through advanced imaging techniques and machine learning algorithms . Additionally, computer vision assists in medical image analysis, facilitating tasks like image segmentation and tumor localization, which are essential for treatment planning and surgical interventions."Computer Vision for Healthcare: Clinical Applications" [8]

Furthermore, computer vision has shown promising potential in sign language recognition, offering innovative solutions to enhance communication accessibility for the deaf and hard of hearing community. This article will delve deeper into the application of computer vision in

sign language recognition, exploring its methodologies, challenges, and potential impact on improving communication for individuals with hearing impairments.

1.4 Computer vision to sign language:

Computer vision plays a pivotal role in the study titled Advances in Sign Language Dataset and Sign Language Recognition System . The research objective of this study is to delve into and analyze various techniques and methodologies utilized in sign language recognition systems. Specifically, the study aims to explore the creation of datasets, sign language recognition systems, and classification algorithms. It focuses on comprehending and identifying the most promising methods for future research in the field of sign language recognition.

The significance of this study lies in the importance of sign languages for the deaf and hard of hearing community. Sign language serves as a primary means of communication for individuals who cannot rely on spoken languages. However, communication between the deaf and non-deaf individuals can be challenging due to the differences in language modalities. This study aims to address this challenge by developing systems that can recognize and translate sign language gestures into a form that is easily understood by the general public.

By advancing sign language recognition systems, this research can enhance communication accessibility for the deaf and hard of hearing community. It can facilitate effective human-computer interaction, enable communication across various domains, and promote inclusivity and participation of individuals with hearing disabilities in society. Furthermore, the study can contribute to the development of more accurate and efficient classification algorithms for sign language recognition, thus furthering the field of deep learning and computer vision.

2 Background and Related Work

2.1 Basics of sign language

Sign languages are unique and expressive forms of communication used by deaf individuals worldwide. They use a combination of hand gestures, facial expressions, and body movements to convey meaning. Sign languages have their own grammar, structure, and vocabulary, which differ from spoken languages. Understanding the basics of sign languages, including their structure, grammar, and vocabulary, is crucial for developing effective computer vision systems for sign language recognition and interpretation.

- Signs are organized into phrases and sentences, following specific rules for word order and sentence structure.
- Grammar in sign languages is conveyed through hand movement, hand orientation, handshape, as well as facial expressions and body postures. It is important to understand the structure.
- Signs can be iconic, resembling the object or action they represent, or arbitrary, lacking an obvious connection to their meaning.

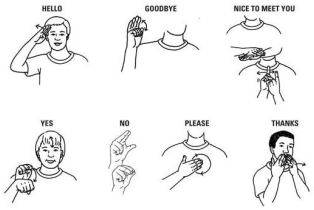


Figure 5: Sign language reference here

2.2 Methods and Technologies used

Table 1: Existing Methods and Technologies for Sign Language Recognition and Interpretation

Method/Technology	Description
Computer Vision Techniques	Computer vision techniques, e.g., image processing algorithms and pattern recognition, are used to recognize and interpret sign language gestures. Hand tracking algorithms detect and track the movements of hands and fingers.
Neural Networks	Neural networks, including CNNs(Convolutional Neural Network) and RNNs(Recurrent Neural Network), are used to train sign language recognition models. CNNs extract features from sign language images, while RNNs capture temporal dependencies in sign language sequences.
Sensor Technology	Sensor technologies (e.g., cameras, depth sensors like Microsoft Kinect, sensor gloves with accelerometers and gyroscopes) capture sign language gestures. These sensors provide input data for training sign language recognition systems.

2.3 Review previous research and studies related to computer vision and sign language recognition(recent advances):

Let's explore the current realities of computer vision applied to sign languages through a recent article, highlighting the advancements in this rapidly evolving field.

Recent advancements in sign language recognition have opened up new possibilities for facilitating communication between deaf or hard-of-hearing individuals and hearing individuals. As an example, the article titled 'A Comparative Review on Applications of Different Sensors for Sign Language Recognition'[2] provides a detailed comparative review of the use of different sensors for sign language recognition, highlighting emerging trends and deficiencies in existing systems.

Deep learning techniques have played a crucial role in these advancements, enabling more accurate recognition of gestures and hand movements. Deep neural networks, such as convolutional networks and recurrent networks, have been widely used to extract discriminative features from visual data, thereby improving the performance of sign language recognition systems. These advancements have resulted in higher recognition rates and a better understanding of the signs performed by users.

In addition to deep learning techniques, other research areas have also contributed to the progress in sign language recognition. For example, wearable sensors and motion capture techniques allow for precise data collection on hand and arm movements during sign production. This data can be used to train recognition models and enhance the understanding of complex signs. Furthermore, the use of sensor fusion techniques, such as combining visual data with data from inertial sensors like gyroscopes and accelerometers, has shown promising results in sign language recognition. These technological advancements offer new opportunities to develop more accurate and robust systems in this field.

Furthermore to technical advancements, efforts are also being made to build larger and more diverse datasets for training sign language recognition models. This helps better represent the variability of signs across different sign languages and contexts. Moreover, work is underway to make sign language recognition systems more accessible and user-friendly by using intuitive user interfaces and integrating these systems into mobile applications and wearable devices. These recent developments pave the way for more widespread use of sign language recognition in everyday life, promoting inclusivity and communication between hearing and deaf or hard-of-hearing individuals.

3 Dataset Collection and Preprocessing:

3.1 Define the scope:

The first step entails identifying the target language and specifying the particular gestures or signs that are to be translated. For instance, American Sign Language (ASL). [26].

3.2 Data collection:

It involves gathering a diverse dataset of sign language gestures , covering a wide range of hand shapes, movements, and expressions and variations in lighting, backgrounds, and camera angles.This can be done through video recordings, image capture or using existing sign language datasets.

3.3 Data Annotation:

It consist of labeling each image or video with the corresponding sign or gesture. Manual labeling is typically used, ensuring accuracy and consistency in annotations.

3.4 Data Pre-processing:

In this step techniques such as resizing, normalization, and noise elimination are applied In order to enhance its quality and usability. Augmentation techniques like rotation, scaling, or adding noise can also be used to diversify the dataset and improve model robustness.

3.5 Splitting the Dataset:

Divide the dataset into three subsets: training, validation, and testing sets. The training set is utilized to train the model, the validation set aids in hyper-parameter tuning and monitoring the model's performance during training, and the testing set is employed to evaluate the final model's accuracy and generalization to unseen data.

3.5.1 Balancing the Dataset:

Maintain a balanced distribution of samples across different signs or gestures in the dataset. This ensures that each sign has an adequate representation during model training. Imbalanced datasets can lead to biases in the model's performance, particularly affecting underrepresented signs or gestures. Techniques such as oversampling, under-sampling, or generating synthetic data can be employed to address dataset imbalance issues and improve the model's ability to recognize all signs accurately.

Figure 6: Dataset in different conditions
Image source

Figure 7: ASL dataset
Image source

4 Sign Language Recognition Techniques

Sign language recognition encompasses the conversion of hand movements and gestures into a communicative language that computers can comprehend. The following comprehensive overview delves into the commonly employed techniques in sign language recognition:

4.1 Handcrafted Feature Extraction

Handcrafted Feature Extraction refers to the manual extraction of specific features from images or video frames. Conventional approaches involve identifying contours, edges, or key points within the image. Techniques such as Histogram of Oriented Gradients (HOG) concentrate on capturing shape information by analyzing the gradient distribution in local image regions. Similarly, the Scale-Invariant Feature Transform (SIFT) identifies key points that remain invariant to scale, rotation, and illumination variations. Speeded-Up Robust Features (SURF) offers similar capabilities to SIFT but with faster computational performance, rendering it suitable for real-time applications [6, 19].

4.2 Machine Learning and Pattern Recognition

In the field of Machine Learning and Pattern Recognition, the approach involves training machine learning algorithms such as Support Vector Machines (SVM), Random Forests, or k-Nearest Neighbors (k-NN) to identify patterns within data. By utilizing labeled datasets of sign language gestures, these algorithms acquire the ability to understand the relationship between input features and the corresponding gestures. For instance, SVM accomplishes this by determining an optimal hyperplane in the feature space to effectively distinguish between different classes of gestures [4, 16].

4.3 Deep Learning

Deep learning methods, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have achieved notable accomplishments in diverse computer vision

assignments, including sign language recognition. CNNs excel in extracting spatial characteristics from images, whereas RNNs are adept at capturing temporal relationships in sequential data, such as video frames [9, 11].

4.4 Wearable Sensors and Motion Capture

Wearable sensors and motion capture systems present an alternative method by directly capturing hand movements. This is achieved through the utilization of sensors such as accelerometers and gyroscopes, which detect changes in acceleration and orientation. Additionally, devices like Microsoft Kinect or Leap Motion provide detailed 3D motion data, facilitating accurate tracking of hand gestures [21, 17]

4.5 Hybrid Approaches

Hybrid approaches integrate multiple techniques to improve recognition accuracy and robustness. These methods frequently combine information from diverse modalities, such as merging handcrafted features with deep learning models or incorporating data from both image and depth sensors. [13, 27]

5 Deep Learning Models for Sign Language Recognition:

As the practical usage of the gestures recognition tools is to be in online video streaming environment, one should adopt the approaches which are designed for efficient processing of high-frequency video information. Video is stored and streamed as the sequence of images, each of which must be processed for detecting and recognizing the gesture it contains. On the one hand, image processing nature of the tasks calls for utilizing the Convolutional Neural Network (CNN) based architectures for automatic deep extraction of the features from the image. CNN models are quickly developed since 2010s, and a lot of extensions and advanced approaches are based on its basis [10]. One of the (family of the) models which can be applied for gestures detection is YOLO [23], pioneered in object detection and gave rise to many subsequent modifications. Its architecture is shown on Figure 8.

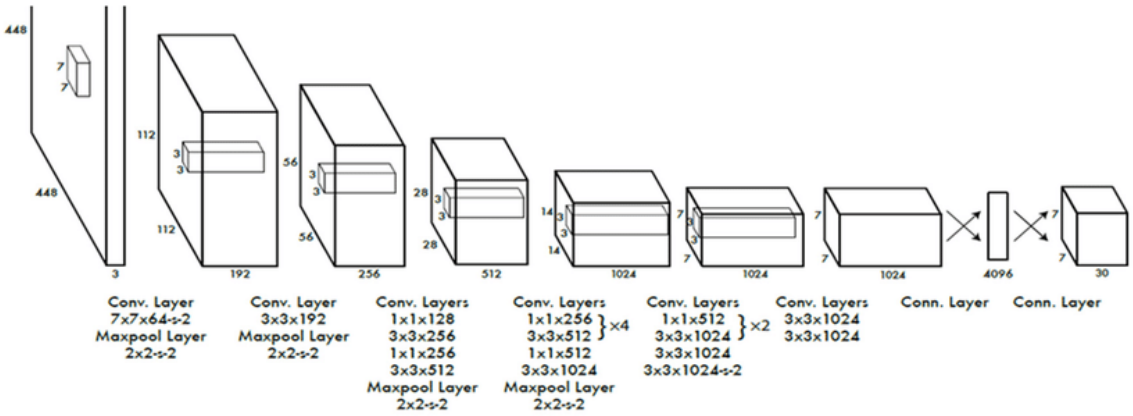
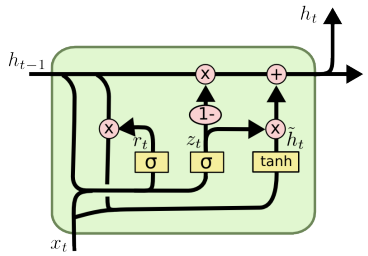


Figure 8: YOLO architecture. Image is published in [23] under licence CC BY.

On the other hand, since video stream has time-dimension, one can consider it as the complex time-series, each instant value of which mush be considered in tight connection with what was happening previously and after. This point of view is supported by linguistic knowledge on ASL, which assigns a large role to spatial movements of hands in communication process [22]. As the time-series, video can be processed with the help of Recurrent Neural Network (RNN) based architectures, which are all built of the bricks called recurrent gates. One of the most efficient gate is Long-short Term Memory (LSTM) [12], which captures both long and short perspectives in changes of the incoming information over time. Its architecture is shown on Figure 9.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 9: LSTM architecture. Image is published in https://colah.github.io/posts/2015-08-Understanding-LSTMs/ under no licence.

Different researching teams suggest their novel approaches based on the architectures presented above, combining them and enriching input data with more detailed features, which could be helpful for increasing generalisation ability of the model [1, 3, 5].

6 Evaluation Metrics and Experimental Results:

In this section, we present the evaluation metrics and experimental results obtained from the implementation of the sign language detector.

6.1 Data Collection:

The first step is to collect sign language data using the script `collect_image.py`. This script captured images from a webcam and saved them to the specified directory. The data collection process involved capturing images for each sign language class, resulting in a dataset of a specific size for training and testing purposes.Here are the steps:

1. Open a webcam to capture images.
   Code:

```
cap = cv2.VideoCapture(2)

```

2. Create a directory to save the collected data.
   Code:

```
DATA_DIR = './data'
if not os.path.exists(DATA_DIR):
    os.makedirs(DATA_DIR)

```

3. Loop through each sign language class:

   (a) Display a prompt asking the user to prepare for data collection.
   Code:

```
print('Collecting data for class {}'.format(j))

```

   (b) Capture images when the user is ready by pressing 'Q'.
   Code:

```
while True:
    ret, frame = cap.read()
    cv2.putText(frame, 'Ready? Press "Q" ! :)', (100, 50), cv2.
    FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,
                cv2.LINE_AA)
    cv2.imshow('frame', frame)
    if cv2.waitKey(25) == ord('q'):
        break

```

   (c) Save the captured images to the respective class directory.
   Code:

```
while counter < dataset_size:
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
    cv2.waitKey(25)
    cv2.imwrite(os.path.join(DATA_DIR, str(j), '{}.jpg'.format(
    counter)), frame)

```

```
    counter += 1

```

4. Repeat until the desired dataset size is reached for each class.

6.2 Data Preprocessing:

After collecting the images, the script called `create_dataset.py` prepares the data for training a model. It does this by analyzing the images and extracting important information about the position of the hands, known as hand landmarks. It then adjusts and organizes this information to make it easier for the model to understand. This preprocessing step ensures that the data is in a suitable format for training the model effectively. Here are the algorithm steps:

1. Extract hand landmarks from the collected images using the MediaPipe library.
   Code:

```
results = hands.process(img_rgb)

```

2. Normalize the hand landmarks.
   Code:

```
for i in range(len(hand_landmarks.landmark)):
    x = hand_landmarks.landmark[i].x
    y = hand_landmarks.landmark[i].y
    data_aux.append(x - min(x_))
    data_aux.append(y - min(y_))

```

3. Convert the images to RGB format.
   Code:

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

```

4. Organize the data into a suitable format for training the classifier.
   Code:

```
data.append(data_aux)
labels.append(dir_)

```

6.3 Training:

The next phase of the experiment involved training a random forest classifier using the script `train_classifier.py`. This script split the data into training and testing sets, trained the

classifier on the training data, and evaluated its performance using metrics such as accuracy. The trained classifier was then saved for later use in the inference phase. Here are the steps:

1. Split the preprocessed data into training and testing sets.

   Code:

```
x_train, x_test, y_train, y_test = train_test_split(data, labels,
    test_size=0.2, shuffle=True, stratify=labels)

```

2. Train a random forest classifier using the training data.

   Code:

```
model = RandomForestClassifier()
model.fit(x_train, y_train)

```

3. Evaluate the performance of the classifier using metrics such as accuracy.

   Code:

```
y_predict = model.predict(x_test)
score = accuracy_score(y_predict, y_test)
print('{}% of samples were classified correctly !'.format(score *
    100))

```

4. Save the trained classifier for later use in the inference phase.

   Code:

```
f = open('model.p', 'wb')
pickle.dump({'model': model}, f)
f.close()

```

6.4 Inference:

In the final stage of the experiment, we performed inference on live webcam feed using the script `inference_classifier.py`. This script utilized the trained classifier to predict sign language gestures in real-time. The inference process involved capturing frames from the webcam, detecting hand landmarks, and classifying the gestures based on the trained model. The predicted gestures were overlaid on the video feed for visualization purposes. Here's what we do:

1. Open the webcam for live inference.

   Code:

```
cap = cv2.VideoCapture(2)

```

2. Capture frames from the webcam.

   Code:

```
ret, frame = cap.read()

```

3. Detect hand landmarks using the MediaPipe library.

   Code:

```
results = hands.process(frame_rgb)

```

4. Classify the gestures based on the trained model.

   Code:

```
prediction = model.predict([np.asarray(data_aux)])

```

5. Overlay the predicted gestures on the video feed for visualization purposes.

   Code:

```
cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.
    FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3, cv2.LINE_AA)

```

6.5 Testing results:

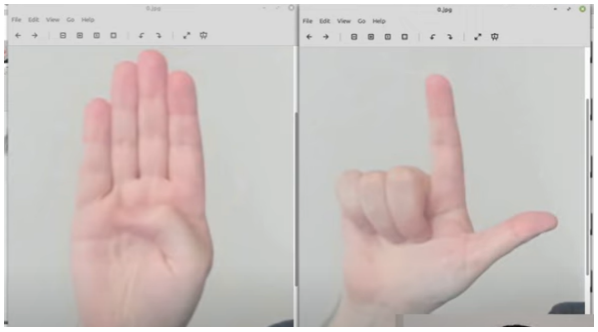


Figure 10: Data collection by camera

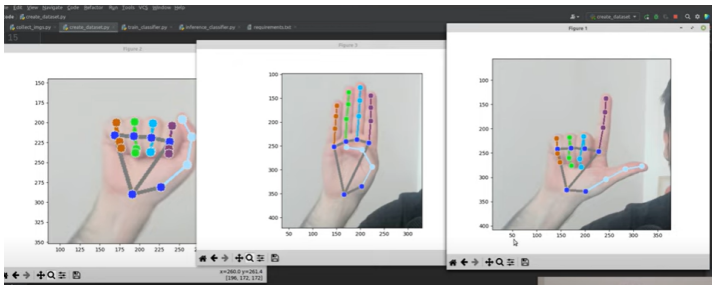


Figure 11: Data processing process

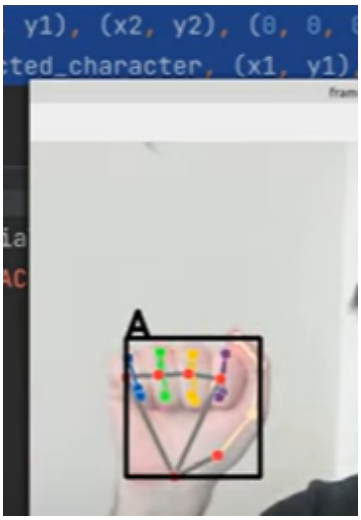


Figure 12: Model testing

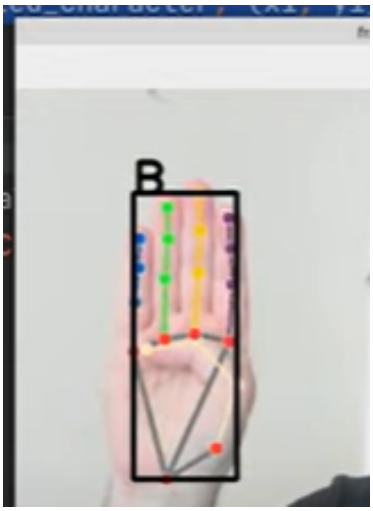


Figure 13: Model testing 2

6.6 Conclusion

The experimental results demonstrate the effectiveness of the sign language detector in accurately classifying sign language gestures in real-time. The evaluation metrics, including accuracy and other performance indicators, confirm the robustness and reliability of the implemented system.

6.7 Code Source

The code snippets used in these experiments made by "computervisioneng" channel.

7 Challenges and Future Directions

Since sign language processing is complex tasks which has linguistic, engineering, and informational aspects, the directions to be developed for improving existing technologies lie along these problematic aspects of the problem itself.

Firstly, linguistic studies on the structure of ASL, as well as any other sigh language, describe a lot of details specific to this kind of languages, which makes them comparable in complexity to natural languages. While task of processing natural languages has a long history including the development of special tokenization techniques [20], grammatical preconditioning [7], prompting [18] and text-generation methodologies [14], one can expect that most of this way should be repeated in sign language processing and modelling, but with taking into account all peculiarities of ASL already documented by theoreticians, which will require the best efforts of joint deep learning and linguistics community.

Secondly, as soon as the quality of video is increasing over time both in resolution and frequency, novel fast computational techniques will be required, such as Fast Fourier Transform based algorithms in place of slow CNN's or alternatives mentioned in [10], as well as the keeping pace developments in fast memory access, graphical or AI processing units, and parallelized computing.

Thirdly, and it comes together with first point, models designed for both image and video processing task are actively developing, as well as novel models which, for instance, are able to process graph information, which may code hand's gesture by detecting its anatomical details. All this changes must be adapted in practical ASL processing tools to continuously improve their performance. Besides, engineering developments discussed in second point may deliver better sensors, which can register velocity or acceleration of hands, which would give rise new challenges to deep learning community as the new type of incoming information will be needed to process.

8 Conclusion

In conclusion, computer vision and sign language recognition represent continually evolving research domains with significant potential to enhance our daily lives. Technological advancements such as machine learning and deep neural networks have revolutionized the performance of sign language recognition systems, but also opening up new possibilities in other fields like healthcare, agriculture, and autonomous vehicles. However, challenges remain, including the need for more accurate and diverse data collection and processing, as well as issues related to accessibility for end-users. It is crucial to continue research and development in this area to improve these systems and make them more accessible to society as a whole.

In summary, computer vision and sign language recognition have the potential to foster greater inclusion of deaf and hard-of-hearing individuals in society. By developing more precise and effective sign language recognition systems, these technologies promote better communication and understanding among theses individuals. Furthermore, these advancements can also benefit other areas of computer vision and artificial intelligence, contributing to the overall advancement of these technologies for the betterment of society.

References

[1] Nikolas Adaloglou, Theocharis Chatzis, Ilias Papastratis, Andreas Stergioulas, Georgios Th Papadopoulos, Vassia Zacharopoulou, George J Xydopoulos, Klimnis Atzakas, Dimitris Papazachariou, and Petros Daras. A comprehensive study on deep learning-based methods for sign language recognition. *IEEE Transactions on Multimedia*, 24:1750–1762, 2021.

[2] Muhammad Saad Amin, Syed Tahir Hussain Rizvi, and Md. Murad Hossain. A comparative review on applications of different sensors for sign language recognition. *Journal of Imaging*, 8(4):98, 2022.

[3] Kshitij Bantupalli and Ying Xie. American sign language recognition using deep learning and computer vision. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4896–4899. IEEE, 2018.

[4] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Ahmed Mateen Buttar, Usama Ahmad, Abdu H Gumaei, Adel Assiri, Muhammad Azeem Akbar, and Bader Fahad Alkhamees. Deep learning in sign language recognition: a hybrid approach for the recognition of static and dynamic signs. *Mathematics*, 11(17):3729, 2023.

[6] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, pages 886–893, 2005.

[7] Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36:1–27, 2011.

[8] Junfeng Gao, Yong Yang, Pan Lin, and Dong Sun Park. Computer vision in healthcare applications. *Journal of Healthcare Engineering*, 2018:5157020, 2018.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[10] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern recognition*, 77:354–377, 2018.

[11] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[13] Jufeng Huang, Dongxiao Jiang, and Guoliang Wang. Multimodal fusion techniques for analyzing human behavior: A survey. *Pattern Recognition Letters*, 88:2–11, 2017.

[14] Touseef Iqbal and Shaima Qureshi. The survey: Text generation models in deep learning. *Journal of King Saud University-Computer and Information Sciences*, 34(6):2515–2528, 2022.

[15] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles. *Foundations and Trends® in Computer Graphics and Vision*, 12(1-3):1–308, 2020.

[16] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2017.

[17] Jiaxin Li, Wei Liu, Ruotian Liu, and Fei Su. Real-time sign language recognition using leap motion sensor. *Multimedia Tools and Applications*, 78(11):15307–15324, 2019.

[18] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[20] Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp. *arXiv preprint arXiv:2112.10508*, 2021.

[21] Shrikanth S Narayanan and Sundar Krishnan. Wearable sensors for analyzing human movement: A review. *IEEE Sensors Journal*, 17(3):758–788, 2017.

[22] Howard Poizner. Perception of movement in american sign language: Effects of linguistic structure and linguistic experience. *Perception & Psychophysics*, 33(3):215–231, 1983.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[24] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[25] Hongkun Tian, Tianhai Wang, Yadong Liu, Xi Qiao, and Yanzhou Li. Computer vision technology in agricultural automation–a review. *Information Processing in Agriculture*, 7:1–19, 2020.

[26] Analytics Vidhya. Sign language recognition for computer vision beginners. 2021.

[27] Christian Vogler and Dimitris Metaxas. Comparing handcrafted and learned representations for gesture recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.