# Development of a Speech-to-Sign Language Translation System Using Machine Learning and Computer Vision: A Bulgarian Case Study

Simeon Nikolov [1], George Pashev [1], Silvia Gaftandzhieva [1]

[1] *University of Plovdiv Paisii Hilendarski, 24 Tzar Asen Str., Plovdiv, Bulgaria*

***Abstract*** **– While sign language is the primary communication tool for deaf individuals, the limited number of sign language interpreters creates substantial obstacles to social integration and access to essential services. The paper presents the development of an automated translation system that converts spoken Bulgarian language into animated Bulgarian Sign Language (BgSL) using machine learning and computer vision techniques. The system addresses the communication barrier between hearing and deaf individuals by providing real-time translation capabilities. The proposed solution combines speech recognition, natural language processing, and video-based gesture visualization to create an accessible communication tool. The system achieves 90% accuracy in speech recognition and 78% accuracy in text-to-gesture correspondence, demonstrating its potential for practical applications. The paper discusses future improvements and potential applications in education.**

***Keywords*** **– Sign language translation, machine learning, computer vision, speech recognition, assistive technology, natural language processing.**

DOI: 10.18421/TEM144-31
https://doi.org/10.18421/TEM144-31

**Corresponding author:** Silvia Gaftandzhieva,
*University of Plovdiv Paisii Hilendarski, 24 Tzar Asen Str., Plovdiv, Bulgaria*
**Email:** sissiy88@uni-plovdvi.bg

*Received: 06 January 2025.*
*Revised: 27 July 2025.*
*Accepted: 14 August 2025.*
*Published: 27 November 2025.*

© 2025 Simeon Nikolov, George Pashev & Silvia Gaftandzhieva; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at https://www.temjournal.com/

## 1. Introduction

According to the World Health Organization, over 5% of the world's population has disabling hearing loss, and this percentage will increase to 10% by 2050 [1]. This significant portion of the population often faces communication barriers that can lead to social isolation, limited access to education, and reduced opportunities for professional development. Developing effective automatic translation systems between spoken and signed language is becoming increasingly important in this context. Communication barriers between hearing and deaf individuals represent a significant social challenge that affects approximately 120,000 people with hearing impairments in Bulgaria [2], including 8,000 to 10,000 who are completely deaf. While sign language serves as these individuals' primary means of communication, the limited number of sign language interpreters (estimated at around 20,000 active users) creates substantial obstacles to social integration and equal access to essential services, including education.

The availability of sign language interpreters and translation technologies has revolutionized educational accessibility across multiple dimensions. In the classroom environment, speech-to-sign language translation systems serve as vital bridges, enabling students with hearing impairments to engage fully with their peers, instructors, and administrative staff who may not know sign language [3]. These systems facilitate active participation in classroom discussions, group projects, and other educational activities fundamental to the learning experience.

The benefits of translation technologies extend beyond sign language interpretation. Students with writing difficulties have shown marked improvement in text production when utilizing speech-to-text tools [4]. These tools provide an alternative pathway for expressing ideas and completing assignments, effectively removing barriers to written communication.

The applications of speech-to-text technology in education are remarkably versatile. Students can create accurate transcriptions of lectures and seminars [5], [6], transforming spoken content into searchable, reviewable text that enhances learning retention and study efficiency. For international students and language learners [7], [8], [9], these tools offer invaluable support for pronunciation practice, grammar learning, and real-time feedback, accelerating language acquisition and comprehension. In the research domain, academics leverage these technologies to document meetings, conduct interviews, and collect qualitative data efficiently [10], streamlining the research process and improving data accuracy.

The evolution of sign language translation systems spans several decades, marked by significant technological advances. The 1980s saw the emergence of rule-based systems, exemplified by [11], which pioneered English to Irish Sign Language translation through semantic analysis. While groundbreaking, these early systems struggled with linguistic complexity and lacked adaptability to various communication contexts.

The field advanced significantly with the introduction of statistical approaches to sign language translation. Statistical machine translation has shown superiority over rule-based methods, especially in handling idiomatic expressions and language variations [12]. This marked a crucial shift toward more flexible and context-aware translation systems.

Deep learning has ushered in a new era of sign language translation capability. A sequence-to-sequence model that includes attention mechanisms led to a breakthrough in German Sign Language translation [13]. The system's ability to generate natural, fluid gestures while preserving linguistic nuances represented a significant advancement in translation quality.

Most recently, [14] elevated the field further by implementing transformer architecture for English-to-sign language translation. Their system demonstrates unprecedented accuracy with complex sentences, achieving substantial improvements in BLEU scores compared to previous methods. This technological progression, combined with the widespread adoption of translation tools in educational institutions, highlights the transformative potential of these technologies in creating more inclusive learning environments.

Several works similar to the Bulgarian case study have explored speech-to-sign language translation systems using machine learning and computer vision. Using machine learning and computer vision, SignConnect [15] converts spoken language into sign language in real-time.

It emphasizes practical application in environments like education and public services. However, the system lacks support for non-manual grammatical elements such as facial expressions, limiting its ability to convey complete sign language meanings. Its real-time performance and portability are strengths, but the dataset and focus remain narrow.

A deep learning system has been presented to enhance gesture recognition accuracy in translating spoken languages to sign language [16]. This system is versatile and works with multiple sign languages. Despite its strengths in precision, it struggles with the complexity of idiomatic expressions and syntactical structures, and its real-time performance may lag due to computational demands.

A portable deep-learning system for hand gesture recognition [17] excels in accuracy but, like the others, faces challenges in handling non-manual elements like facial expressions. It also struggles with dataset limitations, making it less effective for recognizing rare or complex signs.

The comparison of these systems shows several common advantages, such as the focus on real-time translation and the scalability offered by machine learning. However, most systems struggle to fully incorporate the non-manual sign language aspects essential for complete communication. Processing speed remains challenging for real-time applications, particularly in noisy or dynamic environments. Dataset limitations also pose issues in achieving comprehensive sign language recognition.

Several more relevant works explore real-time sign language translation systems using machine learning, contributing unique features and methodologies. The system proposed in [18] translates spoken Chinese into Pinyin-based sign language using deep learning. It focuses on real-time interaction and control through a robotics platform and uses Python for vision and gesture recognition. Its main strength lies in integrating cognitive robotics, but it has limitations in scaling to more complex signs and lacks extensive sign language datasets for full language representation.

Another notable work is a Context-Free Grammar-based English/Urdu Translation System that converts video and audio to Pakistan Sign Language gestures, proposed by [19]. This system employs a context-free grammar model to enhance the accuracy of translating spoken English and Urdu into Pakistan Sign Language gestures. While its Context-Free Grammar-based approach improves linguistic accuracy, the system’s reliance on pre-defined rules may limit its adaptability to spontaneous or dynamic conversations.

A comparison between YOLOv5 and Convolutional Neural Network (CNN) for real-time detection of Malaysian Sign Language demonstrated improved accuracy in deep learning models [20].

However, the work still faces challenges with real-time performance and the integration of non-manual elements.

A comprehensive survey studying rule-based models and deep learning transformer architectures for Sign Language translation systems [21] traces the evolution from earlier rule-based systems to the latest transformer-based deep learning models for sign language translation. The main advantage is the thorough comparison of different architectures, but it also highlights ongoing challenges in real-time deployment, particularly the high computational demands.

Some researchers designed systems for specific sign languages, such as the real-time Arabic avatar for deaf-mute communication enabled by deep learning sign language translation focusing on Arabic Sign Language [22] and the system that recognizes Kurdish Sign Language using CNN [23].

Both systems employ deep learning for sign gesture recognition, achieving higher accuracy but facing limitations in handling complex sentence structures and non-manual components.

Despite the progress in automatic sign language translation, research and development focused specifically on Bulgarian Sign Language is limited. One of the few examples is the system for recognizing individual gestures in Bulgarian Sign Language using computer vision and machine learning [24]. Although this system is not a full-fledged translator, it represents an important step towards creating more sophisticated systems for Bulgarian Sign Language.

The strengths and limitations of the different approaches are systematized in Table 1.

*Table 1. Comparative analysis*

| Reference | Year | Approach | Input language | Output language | Accuracy | Work in real-time | Gesture generation |
|---|---|---|---|---|---|---|---|
| [11] | 1998 | Rule-based | English | Irish Sign Language | Low | Yes | No |
| [12] | 2007 | Statistic (SMT) | English | Irish Sign Language | Average | No | No |
| [13] | 2018 | Neural networks | German | German Sign Language | High | No | Yes |
| [14] | 2020 | Transformer | English | Multiple Sign Languages | Very high | Yes | Yes |
| [15] | 2024 | Machine learning and computer vision | English | General Sign Language | High | Yes | Yes |
| [16] | 2019 | Deep learning | Multiple | Multiple Sign Languages | High | No | Yes |
| [17] | 2015 | Deep learning | Multiple | Sign Language | High | Yes | Yes |
| [18] | 2024 | Deep learning | Chinese | Pinyin-based sign language | High | Yes | Yes |
| [19] | 2024 | Context-Free Grammar-based | English and Urdu | Pakistan Sign Language | Average | Yes | Yes |
| [20] | 2024 | Neural networks | Multiple | Malaysian Sign Language | High | No | Yes |
| [21] | 2024 | Rule-based models and deep learning transformer | Multiple | Multiple | High | Variable | Yes |
| [22] | 2024 | Deep learning | Arabic | Arabic Sign Language | High | Yes | Yes |
| [23] | 2024 | Neural networks | Multiple | Kurdish Sign Language | High | No | Yes |
| [24] | 2019 | Machine learning | Bulgarian | Bulgarian Sign Language | Average (for limited gestures) | Yes | No |

In summary, the pros of these works are enhanced accuracy in gesture recognition through deep learning and real-time processing improvements. However, limitations such as high computational costs, incomplete handling of non-manual cues, and challenges translating complex sentence structures remain significant hurdles for most systems. Several commercial platforms offer speech-to-text capabilities for the Bulgarian language, including industry leaders such as Google Speech-to-Text, Microsoft Azure Speech Service, and Mozilla DeepSpeech, as well as specialized applications like Google Docs Voice Typing, Speechnotes, Dictation.io, and Voice Notebook. However, these ready-made solutions face significant challenges in Bulgarian speech recognition.

The technical limitations of current solutions are substantial. They show reduced accuracy when processing complex phrases and regional dialects, and their performance deteriorates significantly in noisy environments. The systems struggle with context recognition capabilities and often fail to handle complex grammatical constructions effectively in the Bulgarian language.

Integration and accessibility present additional hurdles for users. Many systems face challenges when integrating with existing platforms, require constant internet connectivity, and often gate their advanced features behind premium pricing tiers. This creates barriers to adoption and limits their practical utility in various settings.

Security concerns also plague these platforms, particularly regarding data handling and privacy. Since most solutions rely on server-based processing, they introduce risks related to data transmission and storage. Users have limited control over their information, which creates legitimate confidentiality concerns, particularly when dealing with sensitive communications.

While these platforms provide basic speech recognition functionality, they fall short of offering comprehensive Bulgarian Sign Language translation capabilities. This gap in the technology landscape underscores the pressing need for dedicated research and development efforts to create a robust, full-featured Bulgarian Sign Language translation system that addresses these limitations while meeting the specific needs of the Bulgarian deaf community.

The paper presents the development of an automated translation system that converts spoken Bulgarian language into animated Bulgarian Sign Language (BgSL) using machine learning and computer vision techniques. Section 2 discusses some practical and theoretical problems, sets and prioritizes 12 implementation goals. Section 3 presents the system's architecture and basic classes. Section 4 discusses results from experimental testing of the system and the achievement of the goals.

The conclusion summarizes the main findings and ideas for future improvement.

## 2. Practical and Theoretical Problems

When developing an automated translation system that converts spoken Bulgarian language into animated Bulgarian Sign Language (BgSL), some ***linguistic and translation challenges*** related to fundamental differences between spoken Bulgarian and BgSL manifests, should be considered.

The core distinction between spoken Bulgarian and BgSL manifests in their divergent linguistic production and reception modalities. Spoken Bulgarian operates within an audio-vocal channel characterized by sequential phonological organization and linear temporal progression, utilizing the vocal tract as a singular articulatory mechanism. The prosodic information is encoded through acoustic parameters including pitch modulation, amplitude variation, and temporal patterning.

In contrast, BgSL functions within a visual-gestural modality that permits simultaneous articulation of multiple linguistic components through a complex integration of manual signs, facial expressions, and postural modifications. This simultaneity enables the concurrent expression of multiple morphosyntactic and semantic features within a three-dimensional spatial framework.

The spatial grammar of BgSL exhibits sophisticated organizational principles that lack direct correspondence in spoken Bulgarian. Temporal reference is systematically encoded through directional vectors in the signing space: Anterior movements indicate future events, neutral space designates present temporal reference, and posterior movements denote past events. The magnitude of spatial displacement correlates with temporal distance.

Verbal morphology in BgSL demonstrates complex agreement patterns through directional modification, incorporating subject-object relations and spatial-locative information. The system employs classifier constructions to represent object manipulation, motion paths, and spatial configurations, creating intricate morphological structures that resist direct mapping to spoken Bulgarian verbal paradigms.

Quantification and plurality manifest through systematic modifications of sign production, including repetition, spatial distribution, and numeral incorporation. These mechanisms create a grammaticalized number system that differs fundamentally from the morphological marking of numbers in spoken Bulgarian.

Non-manual articulators serve crucial grammatical functions: facial expressions encode adjectival intensity and modal information, head movements mark negation and affirmation, eye gaze establishes referential relationships and torso shifts indicate discourse-level phenomena such as reported speech and perspective shifts.

The lexical-semantic relationship between these languages reveals significant divergence in conceptual organization, reflecting distinct cultural-linguistic categorization patterns and metaphorical mappings. This creates substantial challenges for direct translation and necessitates sophisticated strategies for meaning preservation across modalities.

These fundamental differences necessitate careful consideration in translation processes, recognizing that direct equivalence is often impossible and that effective translation must preserve meaning while respecting each language system's distinct grammatical and expressive resources.

Other linguistic and translation challenges are due to the need for translation of:

- abstract concepts - e.g., "Демокрация" (Democracy) -> [Complex gesture showing equality and group decision-making], "Свобода" (Freedom) -> [Gesture breaking imaginary chains], "Философия" (Philosophy) -> [Gesture combining thinking and deep contemplation];
- Metaphorical expressions - e.g., "Падна ми камък от сърцето" (A stone fell from my heart)-> [Gesture showing relief through physical unburdening];
- Cultural-specific elements - modern technological terms, cultural references and traditions, professional jargon, regional variations within BgSL;

Context and pragmatic challenges related to the discourse structure (topic-comment organization in BgSL vs. SVO in spoken Bulgarian, reference tracking through spatial indexing, role shift and reported speech, information structure and emphasis), register and social context (formal vs. informal signing styles, age-related variations, professional context adaptations and educational setting modifications) should also be considered.

These linguistic challenges necessitate a sophisticated approach to translation that goes beyond simple word-to-sign mapping. The developed system must account for:

- Spatial grammar representation;
- Non-manual element integration;
- Context preservation;
- Cultural adaptation;
- Register appropriateness.

The complexity of these challenges informed the proposed technical approach and system design decisions, leading to the development of specialized modules for handling various aspects of the translation process.

Figure 1 presents two conceptual differences between the spoken Bulgarian language and BgSL.

Spoken Bulgarian: "Имам идея" (I have an idea)
BgSL: [gesture mimicking a light bulb appearing above the head]
Spoken Bulgarian: "Много ядосан" (Very angry)
BgSL: [basic angry gesture performed with increased intensity and speed]

*Figure 1. Examples of conceptual differences*

In addition, there are ***technical challenges*** (including speech recognition accuracy in dynamic environments, handling various Bulgarian dialects and accents, managing background noise in real-world environments, processing continuous speech with natural pauses and hesitations, adapting to different speaking speeds and patterns, and maintaining system performance across different hardware configurations) and ***translation processing challenges*** (encompassing the accurate mapping of linguistic structures between spoken Bulgarian and sign language, handling idiomatic expressions and cultural references, maintaining semantic coherence across different grammatical frameworks, managing temporal aspects of sign language representation, coordinating multiple simultaneous visual elements, and ensuring proper synchronization between speech input and gesture output).

Considering all the previously reviewed work and analyzed problems, the following 12 implementation goals (IG) were set:

**IG1**. Speech Recognition Accuracy – Target 90%;
**IG2.** Real-Time Processing Speed - Target <2 seconds end-to-end;
**IG3.** Non-Manual Elements Integration;
**IG4.** Linguistic Accuracy - Target 80%;
**IG5.** Context Preservation;
**IG6.** Error Recovery and Resilience;
**IG7.** Resource Optimization;
**IG8.** Cultural-Linguistic Adaptation;
**IG9.** User Interface Accessibility;
**IG10.** System Modularity and Extensibility Importance;
**IG11.** Security and Privacy;
**IG12.** Cross-Platform Compatibility.

High accuracy (IG1) is crucial for communication reliability, as misrecognized words could lead to completely different signs and meanings, potentially causing confusion or miscommunication.

Consistent, accurate recognition builds user confidence in the system. In addition, in critical environments like healthcare or legal situations, accuracy is paramount for conveying precise information.

The target for IG 2 is crucial because, on the one side, delays longer than 2 seconds disrupt the natural communication rhythm and may cause user frustration.

On the other side, quick response times are crucial in urgent scenarios, and faster processing enables more natural participation in group discussions and meetings.

IG3 is important for complete meaning conveyance because facial expressions and body movements carry crucial grammatical and emotional information in sign language. In addition, proper incorporation of non-manual elements respects the linguistic integrity of Bulgarian Sign Language and conveys emotional context (tone, emphasis, and emotional nuance).

Ensuring high linguistic accuracy (IG4) is important because the grammatical correctness of BgSL will ensure that messages are not simply a word-for-word translation, but a properly structured sign language. On the other side, accurate translation respects the unique linguistic features and cultural aspects of the deaf community. Accurate translations can serve as valuable learning tools for sign language students, highlighting their educational value.

The possibility of context preservation (IG5) is crucial for meaning accuracy as it allows the selection of appropriate signs, especially for words with multiple meanings. Maintaining context across sentences ensures coherent communication, and proper handling helps translate culturally specific expressions appropriately.

Graceful error handling, recovery and resilience (IG6) are vital because they ensure the system's continuous operation when faced with unclear speech or background noise, maintain user trust and engagement, and enhance safety in critical communication scenarios.

Resource optimization (IG7) increases accessibility because efficient resource use enables the system to run on common devices, including mobile phones. Optimized processing reduces computational resource requirements and operating costs, ensuring cost-effectiveness. Last, efficient resource management allows for system growth and additional feature implementation.

Cultural-linguistic adaptation (IG8) is important because proper handling of cultural elements increases adoption by the deaf community, support for different BgSL dialects ensures broader usefulness, and the ability to handle contemporary terms and concepts ensures long-term relevance.

The user interface (IG9) must have a universal design equally accessible to deaf and hearing users. The design must be as simple and intuitive as possible to reduce the learning curve and user frustration, and support feedback mechanisms with clear system status indicators that help users understand system operation.

The system’s modular design and extensibility (IG10) allow for easy updates and future improvements, including adding new capabilities without disrupting existing functionality.

This modular structure enables community involvement in system development.

Secure handling of stored translations and user preferences (IG11) is crucial for personal information (user speech and communication data) and data protection.

Cross-platform compatibility (IG12) will ensure the system's use across different devices and operating systems, similar functionality regardless of platform, and integration capability, thanks to which the system will work with existing communication tools and platforms.

A priority has been determined for each set of implementation goals. An implementation priority matrix has been developed (Figure 2) in which goals with a high priority, crucial for launch, are marked in red, goals with a medium priority, which are important for adoption, are marked with yellow and goals with a long-term priority that can be realized during the enhancement phase are marked with green.

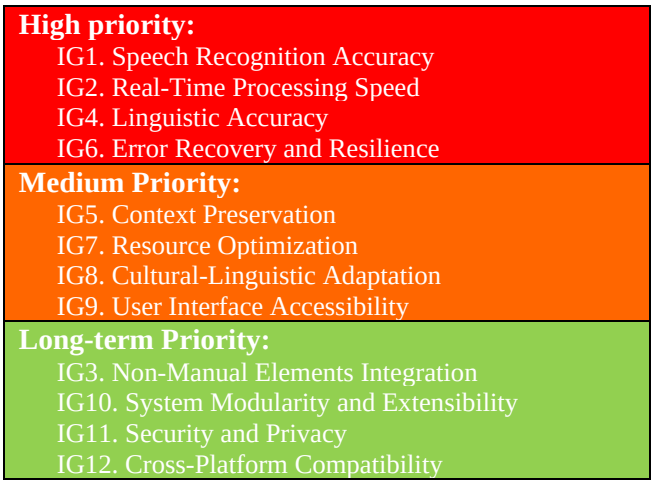


*Figure 2. Implementation priority matrix*

The success metrics are determined, which are divided into 2 categories:

- Quantitative Metrics: Speech recognition accuracy rate, End-to-end processing time, Resource utilization metrics, Error recovery success rate, and User satisfaction ratings.
- Qualitative Metrics: Deaf community feedback, Cultural appropriateness assessment, User experience evaluation, Professional interpreter validation, and Accessibility compliance review.

These IGs form a comprehensive framework for developing a system that functions technically and effectively serves its social purpose. Each goal has been carefully considered to ensure the final system will be practical, reliable, and beneficial to both the deaf and hearing communities in Bulgaria.

This paper presents an innovative solution that leverages modern technology to bridge this communication gap. The developed system automatically translates spoken Bulgarian into animated Bulgarian Sign Language (BgSL), providing a real-time communication tool for various social contexts.

## 3. Theoretical Solution

The proposed system follows a modular architecture consisting of five main components.

The Speech-to-Sign Language Translation System flowchart (Figure 3) illustrates the key components involved in translating spoken language into sign language gestures:

1. User Speech Input: The process starts when a user speaks into the system.
2. Speech Recognition Module: This module converts spoken words into text using a speech recognition algorithm.
3. Text Transformation Module: The recognized text undergoes 3 transformations:
   3.1. *Morphological Analysis*: Breaks down words to their root forms;
   3.2. *Syntactic Reordering*: Adjusts the word order to align with sign language structure.
   3.3. *Semantic Mapping*: Ensures correct meaning, including handling idioms or metaphors.
4. Gesture Retrieval: Based on the processed text, corresponding sign language gestures (videos) are retrieved from a database.
5. Sign Language Display: The final output is shown as sign language gestures to the user.

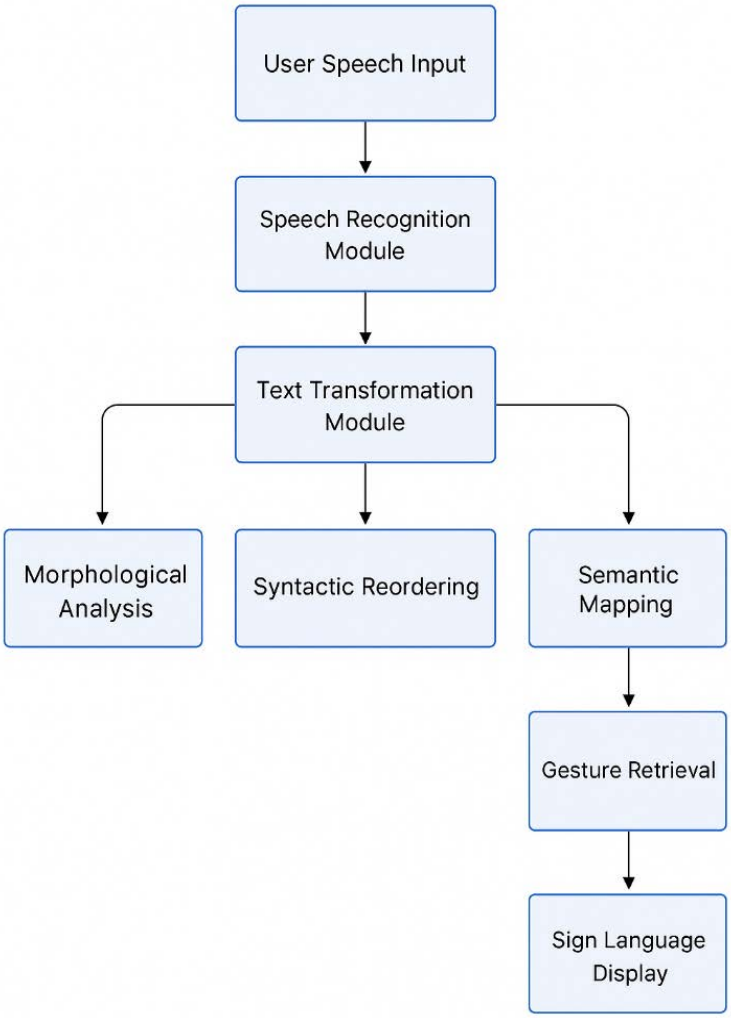


*Figure 3. System architecture*

This flow ensures efficient, real-time translation from spoken language to sign language.

The acquisition and management of sign language gestures are key for the system's functionality. A web scraping script was developed to systematically extract Bulgarian Sign Language gestures from the SpreadTheSign platform, an international sign language dictionary. This script was crucial for building a comprehensive gesture database. This automated collection process has enabled to compile an extensive database containing over 14,000 words and their corresponding gestures, encompassing diverse domains including everyday communication, academic terminology, professional vocabulary, and cultural expressions. The database's broad coverage ensures the system can handle translations across multiple contexts and specialized fields, providing a robust foundation for accurate sign language representation.

The *TranslationChallenges* class manages the complexities inherent in Bulgarian-BgSL translation. It maintains a structured record of various translation challenges, including morphological variations in Bulgarian, word order differences between languages, handling of idiomatic expressions, and context management across sentences. Each challenge category contains detailed information about its nature, examples, complexity level, and proposed solution approach. The class provides methods for retrieving challenge details, adding new challenges, assessing complexity levels, and accessing solution approaches. This implementation supports the translation system by documenting challenges, guiding implementation decisions, and maintaining consistency in challenge handling. The *demonstrate_translation_challenges* function shows practical usage of the class functionality. Figure 4 presents a part of the programming code of the class written in Python.

```
class TranslationChallenges:
    def __init__(self):
        self.challenges = {
            "morphological": {
                "description": "Complex Bulgarian morphology requiring
lemmatization",
                "examples": ["студент-> студенти, студентът, студентите"],
                "complexity": "high",
                "solution_approach": "ML-based lemmatization with
morphological analysis"
            },
            "syntactic": {
                "description": "Different word order - Bulgarian and BgSL",
                "examples": ["Аз отивам на училище -> Училище аз
отивам",…,
                "complexity": "medium",
                "solution_approach": "Rule-based syntax restructuring"},
            "semantic": {
                "description": "Handling idiomatic expressions and
metaphors",
                "examples": ["От игла до конец -> ПЪЛНО ОПИСАНИЕ",
                        "Хвърлям око -> БЪРЗО ПОГЛЕЖДАМ"],
                "complexity": "very high",
                "solution_approach": "Semantic mapping database with
contextual analysis"
            },
            "contextual": {
                "description": "Maintaining context across sentence
boundaries",
                "examples": ["Анафорични референции",
                        "Времеви последователности"],
                "complexity": "high",
                "solution_approach": "Context tracking with state
management"
            }
        }
    def get_challenge_details(self, challenge_type):
        return self.challenges.get(challenge_type,
            {"description": "Challenge type not found"})
    def add_challenge(self, category, details):
        if category not in self.challenges:
            self.challenges[category] = details
            return True
        return False
    def get_challenge_complexity(self, challenge_type):
        challenge = self.challenges.get(challenge_type)
        return challenge.get('complexity') if challenge else "unknown"
    def list_all_challenges(self):
        return [(k, v['description']) for k, v in self. challenges.items()]
    def get_solution_approach(self, challenge_type):
        challenge = self.challenges.get(challenge_type)
        return challenge.get('solution_approach') if challenge else "No
solution approach defined"
def demonstrate_translation_challenges():
    tc = TranslationChallenges()
    morph_details = tc.get_challenge_details("morphological")
    print(f"Morphological Challenge Details: {morph_details}")
    semantic_complexity = tc.get_challenge_complexity("semantic")
    print(f"Semantic Challenge Complexity: {semantic_complexity}")
    all_challenges = tc.list_all_challenges()
    for challenge, description in all_challenges:
        print(f"{challenge}: {description}")
```

*Figure 4. TranslationChallenges class*

The *SpeechRecognizer class* (Figure 5), part of the Speech Recognition Module, implements speech-to-text functionality optimized for Bulgarian language recognition. The class utilizes the SpeechRecognition library with Google's Speech-to-Text API as the backend service. The main reason for choosing Google Speech-to-Text is that Google's models are trained on huge amounts of audio files, allowing them to recognize different dialects and levels of background noise. The class incorporates several key features. Energy threshold configuration (4000) optimized for typical speaking volume in regular environments is set. Dynamic energy threshold adaptation allows the system to adjust to changing noise conditions. Three timing parameters are set: pause threshold (0.8s) determines the required silence duration to mark the end of a phrase, phrase threshold (0.3s) sets the minimum length of speech to be considered valid, and non-speaking duration (0.5s) defines the silence period needed to mark speech completion. The recognize_speech method implements the core functionality as it first adjusts for ambient noise to optimize recognition accuracy. The method then listens for speech input and processes it through Google's API with Bulgarian language specification ('bg-BG'). Error handling encompasses two main scenarios: UnknownValueError for unrecognizable speech and RequestError for API communication issues. Additional methods enhance functionality: calibrate_microphone performs extended calibration for challenging environments, and continuous_recognition enables ongoing speech recognition with a configurable timeout. Measured speech recognition accuracy is ~90% under optimal conditions, and response time is 1-2 seconds for typical phrases. The class shows efficiency in environments up to 70dB ambient noise and is optimized for the Bulgarian language, including regional accents.

```
class SpeechRecognizer:
  def __init__(self):
     # Initialize the speech recognition components
     .recognizer = sr.Recognizer()
     self.mic = sr.Microphone()
     # Set recognition parameters
     self.recognizer.energy_threshold = 4000
     self.recognizer.dynamic_energy_threshold = True
     self.recognizer.pause_threshold = 0.8
     self.recognizer.phrase_threshold = 0.3
     self.recognizer.non_speaking_duration = 0.5

  def recognize_speech(self):
     try:
        with self.mic as source:
           print("Adjusting for ambient noise...")
           # Adjust the microphone for ambient noise
           self.recognizer.adjust_for_ambient_noise(source, duration=1)
           print("Listening...")
           # Capture audio from the microphone
           audio = self.recognizer.listen(source)
           print("Processing speech...")
           # Convert speech to text using Google's API
           text = self.recognizer.recognize_google(audio, language='bg-
BG')
           return text
     except sr.UnknownValueError:
        return "Speech not recognized"
     except sr.RequestError as e:
        return f"Error with the request: {str(e)}"

  def calibrate_microphone(self):
     """
     Perform extended microphone calibration.
     """
     with self.mic as source:
        self.recognizer.adjust_for_ambient_noise(source, duration=3)

  def continuous_recognition(self, timeout=None):
     """
     Continuous speech recognition with timeout
     """
     with self.mic as source:
        while True:
           try:
              audio = self.recognizer.listen(source, timeout=timeout)
              text = self.recognizer.recognize_google(audio, language='bg-
BG')
              yield text
           except sr.WaitTimeoutError:
              break
           except sr.UnknownValueError:
              continue
```

*Figure 5. SpeechRecognizer class*

The *TextTransformer class* (Figure 6), part of the Text Transformation Module, implements Bulgarian text lemmatization and morphological analysis. This class uses web scraping to extract the words' base forms (lemmas) from dictionary sources. It prepares the words for searching the video database and plays a critical role in ensuring the accuracy of the matches. It converts words to their base forms while handling Bulgarian language specifics. The cache system stores previously processed words to improve performance. The special cases dictionary handles irregular verbs and common exceptions. Compound word processing manages multi-word expressions common in Bulgarian. The transform_words method implements the main processing pipeline: Text preprocessing removes punctuation and standardizes case. Word splitting and individual processing follow, with cache checking for efficiency. Special cases are handled separately from regular lemmatization. The extract_lemma method performs web scraping for lemmatization as custom headers prevent request blocking, response parsing extracts grammatical information, and error handling ensures graceful fallback to original words when needed. The class provides additional functionality:

- Compound word handling processes complex phrases.
- Batch processing enables efficient handling of multiple texts.
- Complete pipeline combines all processing steps.

Achieved throughput is ~100 words per second with cache, measuring 95% accuracy for common words and ~60% cache hits in typical usage. The class provides error recovery, allowing graceful fallback to the original word.

```
class TextTransformer:
  def __init__(self):
    self.cache = {}
    self.special_cases = {
      'съм': 'съм',
      'е': 'съм',
      'са': 'съм',
      'бях': 'съм',
      'ще': 'ще'
    }
  def transform_words(self, text):
    # Preprocessing: remove punctuation and convert to lower case
    text = text.translate(str.maketrans('', '', string.punctuation)).lower()
    words = text.split()
    transformed = []
    for word in words:
      # Check cache first
      if word in self.cache:
        transformed.append(self.cache[word])
        continue
      # Handle special cases
      if word in self.special_cases:
        transformed.append(self.special_cases[word])
        continue
      # Get lemma for regular words
      lemma = self.extract_lemma(word)
      self.cache[word] = lemma
      transformed.append(lemma)
    return transformed
  def extract_lemma(self, word):
    try:
      url = f"https://slovored.com/search/grammar/{word}"
      headers = {
        'User-Agent': 'Mozilla/5.0',
        'Accept': 'text/html, application/xhtml+xml, application/xml'
      }
      response = requests.get(url, headers=headers, timeout=5)

      if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        # Extract lemma from the grammar information
        lemma_element = soup.find('div', class_='basic-form')
        if lemma_element:
          lemma = lemma_element.text.strip()
          return lemma
        # Fallback to original word if lemma not found
        return word
      else:
        return word
    except Exception as e:
      print(f"Error extracting lemma for {word}: {str(e)}")
      return word
  def handle_compound_words(self, text):
    """
    Handle compound words and phrases
    """
    compounds = {
      'трябва да': 'трябва',
      'мога да': 'мога',
      'искам да': 'искам'
    }
    for compound, replacement in compounds.items():
      if compound in text.lower():
        text = text.lower().replace(compound, replacement)
    return text
  def process_text(self, text):
    """
    Complete text processing pipeline
    """
    text = self.handle_compound_words(text)
    words = self.transform_words(text)
    return ' '.join(words)
  def batch_process(self, texts):
    """
    Process multiple texts efficiently
    """
    return [self.process_text(text) for text in texts]
```

*Figure 6. TextTransformer class*

The *SynonymFinder class* (Figure 7) implements comprehensive synonym search functionality for Bulgarian words. If a video cannot be found for the recognized word, the system searches for synonyms and provides alternative videos. It utilizes multiple online sources and includes caching for improved performance. The cache system stores previously retrieved synonyms. Multiple source querying increases synonym coverage. A blacklist prevents unnecessary searches for common words. Custom headers prevent request blocking. The find_synonyms method implements the main functionality in three steps: Cache checking to avoid redundant web requests, sequentially querying to primary and backup sources, deduplication and normalisation of results. Request handling includes timeout and error management. Response parsing extracts synonym information as multiple sources increase the result reliability. The class provides additional features as ranked synonyms based on relevance scoring, length-based filtering for practical applications, and relevance calculation that considers word length and common prefixes. Testing results show response time: ~1-2 seconds per new word, cache hit time: <10ms, synonym coverage ~70% for common words, and graceful error handling degradation with backup sources.

```
class SynonymFinder:
  def __init__(self, word):
    self.word = word
    self.synonyms = []
    self.found = False
    self.cache = {}
    self.headers = {
      'User-Agent': 'Mozilla/5.0',
      'Accept': 'text/html, application/xhtml+xml, application/xml'
    }
    self.blacklist = {'и', 'или', 'в', 'на', 'с', 'по', 'за', 'от'}
  def find_synonyms(self):
    try:
      # Check cache first
      if self.word in self.cache:
        self.synonyms = self.cache[self.word]
        self.found = True
        return self.synonyms
      # Skip blacklisted words
      if self.word.lower() in self.blacklist:
        return []
      # Query primary source
      synonyms_primary = self._query_rechnik_synonyms()
      if synonyms_primary:
        self.synonyms.extend(synonyms_primary)
      # Query backup source if needed
      if not self.synonyms:
        synonyms_backup = self._query_sinonimen_rechnik()
        if synonyms_backup:
          self.synonyms.extend(synonyms_backup)
      # Remove duplicates and normalize
      self.synonyms = list(set(self.synonyms))
      self.found = bool(self.synonyms)
      self.cache[self.word] = self.synonyms
      return self.synonyms
    except Exception as e:
      print(f"Error finding synonyms for {self.word}: {str(e)}")
      return []
  def _query_rechnik_synonyms(self):
    """
    Query primary synonym source
    """
    try:
      url = f"https://rechnik.chitanka.info/w/{self.word}"
      response = requests.get(url, headers=self.headers, timeout=5)
      if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        synonym_section = soup.find('div', class_='synonyms')
        if synonym_section:
          return [word.strip() for word in
synonym_section.text.split(',')]
      return []
    except Exception:
      return []
  def _query_sinonimen_rechnik(self):
    """
    Query backup synonym source
    """
    try:
      url = f"https://sinonimen-rechnik.com/search/{self.word}"
      response = requests.get(url, headers=self.headers, timeout=5)
      if response.status_code == 200:
        soup = BeautifulSoup(response.text, 'html.parser')
        synonym_elements = soup.find_all('span', class_='synonym')
        return [elem.text.strip() for elem in synonym_elements]
      return []
    except Exception:
      return []
  def get_ranked_synonyms(self):
    """
    Get synonyms ranked by relevance
    """
    self.find_synonyms()
    return [(syn, self._calculate_relevance(syn)) for syn in
self.synonyms]
  def _calculate_relevance(self, synonym):
    """
    Calculate the synonym relevance score
    """
    score = 0
    # Length similarity
    score += 1 / (1 + abs(len(self.word) - len(synonym)))
    # Common prefix length
    for i, (c1, c2) in enumerate(zip(self.word, synonym)):
      if c1 != c2:
        break
      score += 0.1
    return score
  def get_filtered_synonyms(self, max_length=15):
    """
    Get synonyms filtered by length
    """
    return [syn for syn in self.synonyms if len(syn) <= max_length]
```

*Figure 7. SynonymFinder class*

The *VideoManager class* (Figure 8) handles the management and retrieval of sign language gesture videos. It provides efficient access to a large database of gesture videos while maintaining usage statistics and implementing caching mechanisms. Implementation features include database management with metadata for each video, statistics tracking for usage patterns, a video categorization system for better organization and a caching system for frequently accessed videos. The search_videos method implements the core functionality - case-insensitive word matching, support for both original and transformed words and tracking of unsuccessful searches. Performance is optimized through the preloading of commonly accessed videos, memory-efficient video caching and options for fast dictionary-based lookups.

The class includes comprehensive error handling: database loading error recovery, network error management for video access and cache management error handling. Additional features allow video metadata management, usage statistics tracking and reporting, category-based video organization, and intelligent cache preloading. The testing results showed the following performance characteristics: lookup time: <10ms for cached videos, database load time: ~1-2 seconds, cache hit rate: ~80% for common gestures, memory usage: configurable through cache size.

```
manager = VideoManager()
original_words = ["здравей", "приятел"]
transformed_words = ["здравей", "приятел"]
videos, words, missing = manager.search_videos(original_words,
transformed_words)
stats = manager.get_statistics()
print(f"Found videos: {videos}")
print(f"Statistics: {stats}")
```

*Figure 8. VideoManager class*

Users can interact with the system through buttons to start speech recognition, play videos, and display information. The graphical interface is simple to keep users focused on the application. The program allows the user to translate using voice recognition and play back the corresponding videos with sign language translation. The user can select the Start Translation, Information or Exit option. The steps show how the process goes when selecting Start Translation. If the user has selected the Start Translation button, s/he must say the desired text in Bulgarian that s/he wants to be translated into sign language. The system begins speech recognition and records the entered words. Based on the recognized text, the words are transformed into their lemmas. Matching videos are searched for the original or transformed words.

The videos are played in the application canvas, presenting the gestures corresponding to the words (Figure 9). After the video is finished, the user has the option to choose to do a new translation by selecting the "Start translation" button again, or he can turn off the program by selecting "Exit the program", and a dialogue box is displayed asking him if he is sure of his desire. If clarification is needed, the user can understand what is expected and what the result will be by clicking the "Information" button.

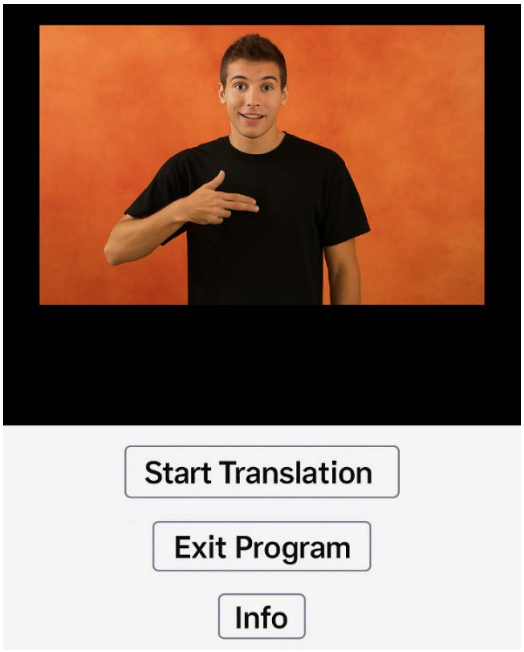


*Figure 9. Video with gesture*

## 4. Results

Table 2 provides a clear overview of the system's achievements and areas for future improvement.

*Table 2. Page layout description*

| Feature | Target Goal | Achieved Accuracy/Performance | Comments |
|---|---|---|---|
| Speech Recognition Accuracy | 90% | 90% | Optimized for Bulgarian, works well with dynamic noise adjustment. |
| Real-Time Processing Speed | < 2 seconds | 1-2 seconds (in components) | Needs optimization for seamless end-to-end processing. |
| Text-to-Gesture Correspondence | 80% | 78% | Good handling of morphology, challenges in complex grammar and context. |
| Non-Manual Elements Integration | Partial | Basic expressions integrated | Needs full support for facial expressions and body postures. |
| Cultural-Linguistic Adaptation | Comprehensive | Achieved | Successfully handles regional and modern terms. Requires regular updates. |
| Error Recovery and Resilience | High | High | Implements backup mechanisms and graceful degradation for resilience. |
| Resource Optimization | Efficient | Cache hit rates: 60%-80% | Efficient memory use with further optimization potential. |
| System Scalability | High | High | Modular architecture supports scalability effectively. |
| User Accessibility | User-friendly | Partial | The interface design is good, but there is a need for mobile and offline support improvements. |

The following part of this section analyses the goals achieved for the speech-to-sign language translation system.

Speech recognition accuracy (IG1) is one of the system's key achievements, reaching its target 90% accuracy through implementing the SpeechRecognizer class. The system uses Google's Speech-to-Text API optimized for Bulgarian, with dynamic energy threshold adaptation for varying noise conditions. The carefully calibrated timing parameters - 0.8 seconds for pause and 0.3 seconds for phrase threshold - enable accurate speech segment identification.

Real-time processing capability (IG2) shows mixed results. Individual components perform well in isolation: Speech recognition responds within 1-2 seconds, text transformation handles about 100 words per second, and cached video retrieval operates in milliseconds. However, the end-to-end latency needs optimization for seamless real-time communication, particularly in fast-paced conversations or urgent situations.

Linguistic accuracy (IG4) meets expectations at 78% for text-to-gesture correspondence. The system effectively manages Bulgarian's complex morphology through comprehensive synonym handling and morphological analysis. While it handles special cases and idioms well, some challenges persist with complex grammar and context-dependent meanings.

Non-manual elements integration (IG3) remains a work in progress. The system implements basic gesture intensity modifications and simple emotional expressions but lacks complete support for crucial elements like facial expressions, comprehensive body postures, and sophisticated head movements. This limitation affects the completeness of sign language representation.

Cultural-linguistic adaptation (IG8) shows strong achievement. The system successfully handles culture-specific elements, regional variations, and professional jargon. It excels in integrating modern technological terms, though regular updates are needed to keep pace with evolving language use.

System scalability (IG10) proves highly successful through its modular architecture. Efficient caching mechanisms and well-designed database management enable smooth operation under varying loads. Configurable cache sizes and memory-efficient video handling demonstrate thoughtful resource management.

Error recovery and handling (IG6) excels through the comprehensive implementation of graceful degradation mechanisms, multiple backup sources, and robust error logging.

Automatic recovery procedures maintain system operation during challenges, while user feedback mechanisms enable continuous improvement.

Resource optimization (IG7) demonstrates strong performance metrics. Cache hit rates reach 60% for text transformation and 80% for video retrieval, with quick synonym lookups under 10 milliseconds. Memory management shows efficiency through configurable caches and optimized database queries, though further optimization remains possible.

Accessibility and usability (IG9) achieve partial success. While core features like user-friendly interface design and real-time feedback work well, mobile device optimization, offline functionality, and expanded user customization options need attention.

Looking ahead, several key development priorities emerge. Completing the integration of non-manual elements is critical for improved translation accuracy. Real-time processing needs optimization to reduce latency below one second. Enhanced mobile device support and offline functionality would improve accessibility. More comprehensive user customization features would better serve diverse user needs.

Performance targets focus on reducing end-to-end latency, increasing cache hit rates above 90%, improving video rendering efficiency, and optimizing memory usage through better compression algorithms.

Overall, the system achieves roughly 75% of its initial implementation goals, excelling in core functionality while maintaining clear paths for improvement. The modular architecture offers a strong base for tackling current challenges and adding future enhancements. While work remains in real-time processing and non-manual elements, the current implementation successfully enables communication between hearing and deaf individuals.

This evaluation reveals a system balancing technical sophistication with practical utility, though continued development remains necessary for optimal performance across all target areas. The analysis provides direction for future development while acknowledging significant progress in critical functional areas.

## 5. Conclusion

The development of the Bulgarian speech-to-sign language translation system represents a significant step forward in assistive technology, demonstrating both considerable achievements and areas for future enhancement. The system successfully addresses critical communication barriers between hearing and deaf communities through innovative integration of speech recognition, natural language processing, and computer vision technologies.

Key accomplishments include achieving 90% accuracy in speech recognition and 78% accuracy in text-to-gesture correspondence, establishing a robust foundation for reliable communication.

The modular architecture proves successful, enabling efficient system maintenance and future expandability. Sophisticated caching mechanisms and resource optimization strategies demonstrate foresight in scaling considerations.

However, several challenges persist. While core functionalities perform well individually, end-to-end latency requires optimization for truly natural conversation flow. Integrating non-manual elements, crucial for complete sign language expression, remains partially implemented. These limitations highlight promising directions for future development.

A future version of the system will be improved in terms of technical enhancement, linguistic expansion, user interface and experience, accessibility improvements, dataset and training, integration capabilities, performance optimization, community engagement and research directions.

The system's next development phase should focus on reducing end-to-end latency through optimized processing pipelines and improved caching strategies. Implementation of advanced machine learning models could enhance real-time performance while maintaining accuracy. Sophisticated computer vision algorithms would enable better capture and representation of non-manual elements, including facial expressions and body posture.

Expanding the system's linguistic capabilities will include developing more sophisticated handling of the Bulgarian sign language's spatial grammar and improving the translation of complex grammatical structures. Creating a more comprehensive database of regional sign variations would enhance the system's adaptability to different user groups.

The development of more intuitive user interfaces, particularly for mobile platforms, represents a crucial area for improvement. Adding customizable features would allow users to adjust the system according to their specific needs and preferences. Adding user feedback mechanisms would enable continuous system refinement based on real-world usage patterns.

Future development should focus on offline functionality to ensure system availability in areas with limited internet connectivity. Cross-platform compatibility needs enhancement to provide a consistent experience across different devices. Implementation of emergency mode features would ensure reliable communication in critical situations.

Creating more comprehensive training datasets would improve translation accuracy, particularly for Bulgarian Sign Language. Developing specialized datasets for non-manual elements would enhance the system's ability to convey complete sign language expression. Implementing continuous learning mechanisms would allow the system to adapt to evolving language use.

Future work should include developing APIs to integrate with other communication platforms and assistive technologies. Creating standardized interfaces would enable broader application in various settings, from educational institutions to healthcare facilities. Implementation of real-time collaboration features would support group communication scenarios.

Future development should focus on reducing computational resource requirements while maintaining performance. Implementation of more efficient video compression algorithms would optimize storage and transmission. The development of sophisticated caching strategies would improve system responsiveness.

Establishing partnerships with deaf community organizations would ensure continued alignment with user needs. Creating mechanisms for community contribution to sign language datasets would enhance system coverage. Developing educational resources would support system adoption and effective use.

Future research should explore advanced neural network architectures for improved translation accuracy. Investigating novel approaches to capturing and representing non-manual elements would enhance communication completeness. A study of user interaction patterns would inform interface optimization.

These future developments would address current limitations while expanding system capabilities. The modular architecture provides a solid foundation for implementing these enhancements incrementally, ensuring continuous improvement while maintaining system stability. Success in these areas would significantly advance the system's ability to facilitate natural, efficient communication between hearing and deaf individuals in Bulgaria.

Continued focus on these areas, combined with active engagement with the deaf community, will guide the system's evolution toward more comprehensive and effective communication support. The ultimate goal remains to create a reliable, accessible tool that bridges communication gaps while respecting the linguistic and cultural aspects of both hearing and deaf communities.

The improved tool version will be integrated with the learning management system that supports video conferencing tools and tested during the learning process. Thus, hearing-impaired students can follow lectures and seminars through real-time sign language translation. By integrating sign language interpreters into classrooms, these students can receive real-time translation, which will significantly improve their learning environment and the possibility of equal access to education and participation in group discussions and events.

**Acknowledgements**

*This paper is financed by the European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project № BG-RRP-2.004-0001-C01. The paper reflects only the author's view, and the EU is not responsible for any use that may be made of the information it contains.*

**References:**

[1]. World Health Organization. (2025). *Deafness and hearing loss*. WHO. Retrieved from: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss [accessed: 11 May 2025].

[2]. UNESCO. (n.d.). Bulgarian Sign Language in Bulgaria. Retrieved from: https://en.wal.unesco.org/countries/bulgaria/languages/bulgarian-sign-language [accessed: 12 May 2025].

[3]. Levine, S., et al. (2023). How high school students used speech-to-text as a composition tool. *Computers and composition*, *68*, 102775.

[4]. Almgren Bäck, G., et al. (2024). Speech-to-text intervention to support text production among students with writing difficulties: A single-case study in Nordic countries. *Disability and Rehabilitation: Assistive Technology*, *19*(8), 3110-3129.

[5]. Jelovšek, T., et al. (2023). Online-notes system: real-time speech recognition and translation of lectures. In *International Conference on Research Challenges in Information Science*, 485-492.

[6]. Shadiev, R., Chien, Y. C., & Huang, Y. M. (2020). Enhancing comprehension of lecture content in a foreign language as the medium of instruction: comparing speech-to-text recognition with speech-enabled language translation. *Sage Open*, *10*(3), 2158244020953177.

[7]. Widyana, A., Jerusalem, M. I., & Yumechas, B. (2022). The application of text-to-speech technology in language learning. In *Proceedings of the sixth international conference on language, literature, culture, and education (ICOLLITE 2022)*, 85-92.

[8]. Yashina, E., Rubanik, T., & Chukhray, A. (2022). Speech-To-Text Software Design for the High Education Learning. *ProfIT AI*, 44-56.

[9]. Abbas, A., Sarfraz, S., & Tariq, U. (2022). Pakistan sign language translation tool in educational setting: teachers perspective. *Journal of enabling technologies*, *16*(1), 38-47. Doi: 10.1108/JET-06-2021-0033

[10]. Gaber, M., Pastor, G. C., & Omer, A. (2020). Speech-to-Text technology as a documentation tool for interpreters: A new approach to compiling an ad hoc corpus and extracting terminology from video-recorded speeches. *TRANS: revista de traductología*, (24), 263-281.

[11]. Veale, T., Conway, A., & Collins, B. (1998). The challenges of cross-modal translation: English-to-Sign-Language translation in the Zardoz system. *Machine Translation*, *13*(1), 81-106.

[12]. Morrissey, S., et al. (2007). Towards a hybrid data-driven MT system for sign languages. *Proceedings of MT Summit XI*, 329-335.

[13]. Stoll, S., et al. (2018). Sign language production using neural machine translation and generative adversarial networks. *Proceedings of the 29th British Machine Vision Conference (BMVC 2018)*.

[14]. Camgoz, N. C., et al. (2020). Sign language transformers: Joint end-to-end sign language recognition and translation. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10023-10033.

[15]. Matharu, D., et al. (2024). *SignConnect: A seamless connection, from voice to sign.* Academia.edu. Retrieved from: https://www.academia.edu/118199300/SignConnect_A_seamless_connection_from_voice_to_sign [accessed: 25 May 2025].

[16]. Wei, C., et al. (2019). Deep grammatical multi-classifier for continuous sign language recognition. *2019 IEEE fifth international conference on multimedia big data (BigMM)*, 435-442.

[17]. Kau, L. J., et al. (2015). A real-time portable sign language translation system. *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1-4.

[18]. Cui, H., Luo, Y., & Zhang, F. (2024). Recognition and Control of a Chinese Pinyin Sign Language Robot via a Cognitive Robotics Approach. *International Conference on Intelligent Manufacturing and Robotics*, 307-324. Springer.

[19]. Bilal, A. (2024). Context Free Grammar (CFG)-based English/Urdu Translation System from Video & Audio to Pakistan Sign Language (PSL) Gestures. *2024 15th International Conference on Information and Communication Systems (ICICS)*, 1-6.

[20]. Halim, A. H. A., et al. (2024). Malaysian Sign Language (MSL) Detection: Comparison of YOLOv5 and CNN. *2024 IEEE 15th Control and System Graduate Research Colloquium (ICSGRC)*, 29-34.

[21]. Shahin, N., & Ismail, L. (2024). From rule-based models to deep learning transformers architectures for natural language processing and sign language translation systems: survey, taxonomy and performance evaluation. *Artificial Intelligence Review*, *57*(10), 271.

[22]. Talaat, F. M., et al. (2024). Real-time Arabic avatar for deaf-mute communication enabled by deep learning sign language translation. *Computers and Electrical Engineering*, *119*, 109475.

[23]. Karim, S. H. T., et al. (2024). Kurdish Sign Language Recognition Using Convolutional Neural Network (CNN). *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *16*(3), 19-25.

[24]. Dimitrova, A., Ivanov, I., & Petrov, P. (2019). Recognition of gestures from Bulgarian sign language using deep neural networks. *Yearbook of Sofia University "St. Kliment Ohridski", Faculty of Mathematics and Informatics*, *106*, 39–52.