CAST: Center of Advanced Software Technologies

Sevak Sargsyan
sevak.sargsyan@rau.am
Head of the system programming department,
Director of the CAST,
Russian-Armenian University

CAST

Agenda

1. Who we are?
2. Our Model
3. Compiler Technologies
4. Software Analysis Technology
5. Natural Language and Speech Processing
6. Autonomous Systems

Our Model

1. University based research center

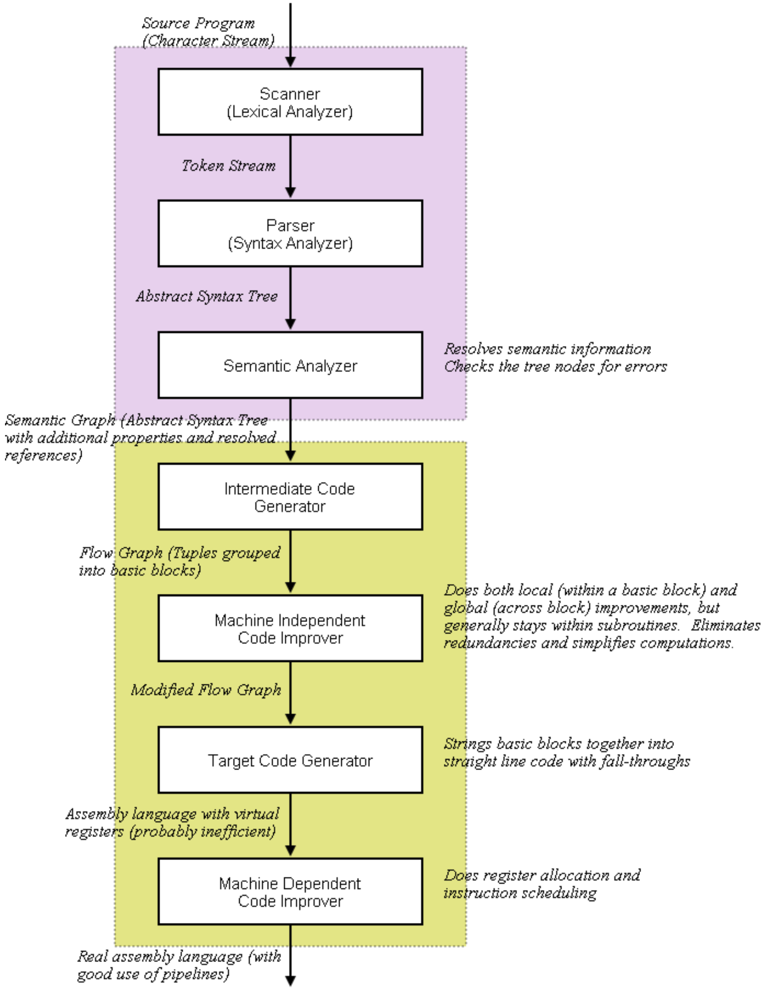
2. Involved in education process:
   - Teach many courses
   - Provide scholarships
   - System programing department is under our supervision

3. Involved in industry projects:
   - Have many collaboration project

4. Involves students in our projects:
   - Provide supervisors for diploma, master and PhD

5. Heavily invest in new research directions (60+ researchers)

Compiler Technologies: Optimizations



Source Program (Character Stream)

Scanner (Lexical Analyzer)

Token Stream

Parser (Syntax Analyzer)

Abstract Syntax Tree

Semantic Analyzer

Resolves semantic information Checks the tree nodes for errors

Semantic Graph (Abstract Syntax Tree with additional properties and resolved references)

Intermediate Code Generator

Flow Graph (Tuples grouped into basic blocks)

Machine Independent Code Improver

Does both local (within a basic block) and global (across block) improvements, but generally stays within subroutines. Eliminates redundancies and simplifies computations.

Modified Flow Graph

Target Code Generator

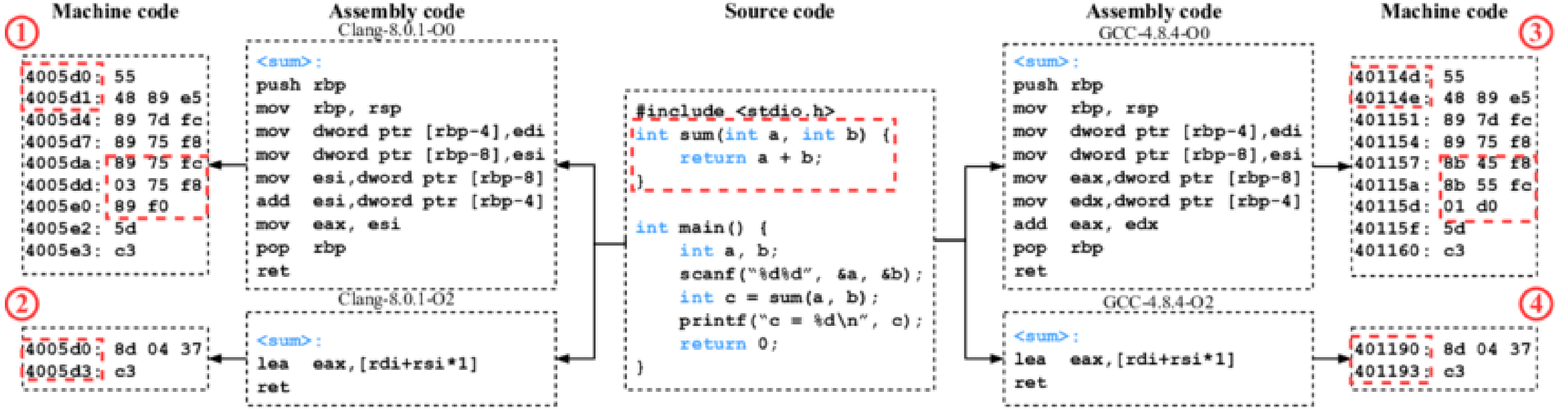
Strings basic blocks together into straight line code with fall-throughs

Assembly language with virtual registers (probably inefficient)

Machine Dependent Code Improver

Does register allocation and instruction scheduling

Real assembly language (with good use of pipelines)



```
4005d0: 55
4005d1: 48 89 e5
4005d4: 89 7d fc
4005d7: 89 75 f8
4005da: 89 75 fc
4005dd: 03 75 f8
4005e0: 89 f0
4005e2: 5d
4005e3: c3
```

```
4005d0: 8d 04 37
4005d3: c3
```

```
<sum>:
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4],edi
mov  dword ptr [rbp-8],esi
mov  esi,dword ptr [rbp-8]
add  esi,dword ptr [rbp-4]
mov  eax, esi
pop  rbp
ret
```

```
<sum>:
lea  eax,[rdi+rsi*1]
ret
```

```
#include <stdio.h>
int sum(int a, int b) {
    return a + b;
}

int main() {
    int a, b;
    scanf("%d%d", &a, &b);
    int c = sum(a, b);
    printf("c = %d\n", c);
    return 0;
}
```

```
<sum>:
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4],edi
mov  dword ptr [rbp-8],esi
mov  eax,dword ptr [rbp-8]
mov  edx,dword ptr [rbp-4]
add  eax, edx
pop  rbp
ret
```

```
<sum>:
lea  eax,[rdi+rsi*1]
ret
```

```
40114d: 55
40114e: 48 89 e5
401151: 89 7d fc
401154: 89 75 f8
401157: 8b 45 f8
40115a: 8b 55 fc
40115d: 01 d0
40115f: 5d
401160: c3
```

```
401190: 8d 04 37
401193: c3
```

Compiler Technologies: Optimizations

1. GCC – Optimal code generation for ARM architecture (patches accepted by community)

2. LLVM – SLP vectorization, instruction scheduling

3. V8 – JIT compilation improvement for «Hot code»

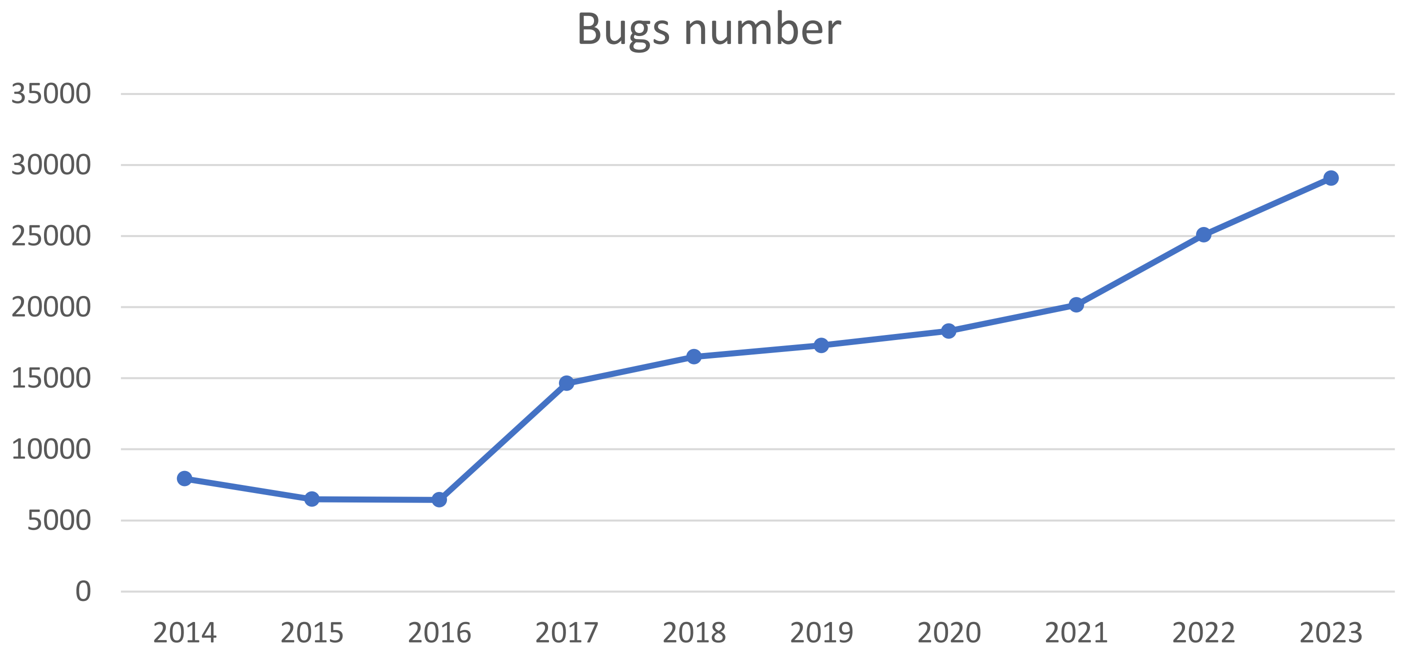
4. V8 – LLVM as backend

5. Webkit – register allocation/ rematerialization

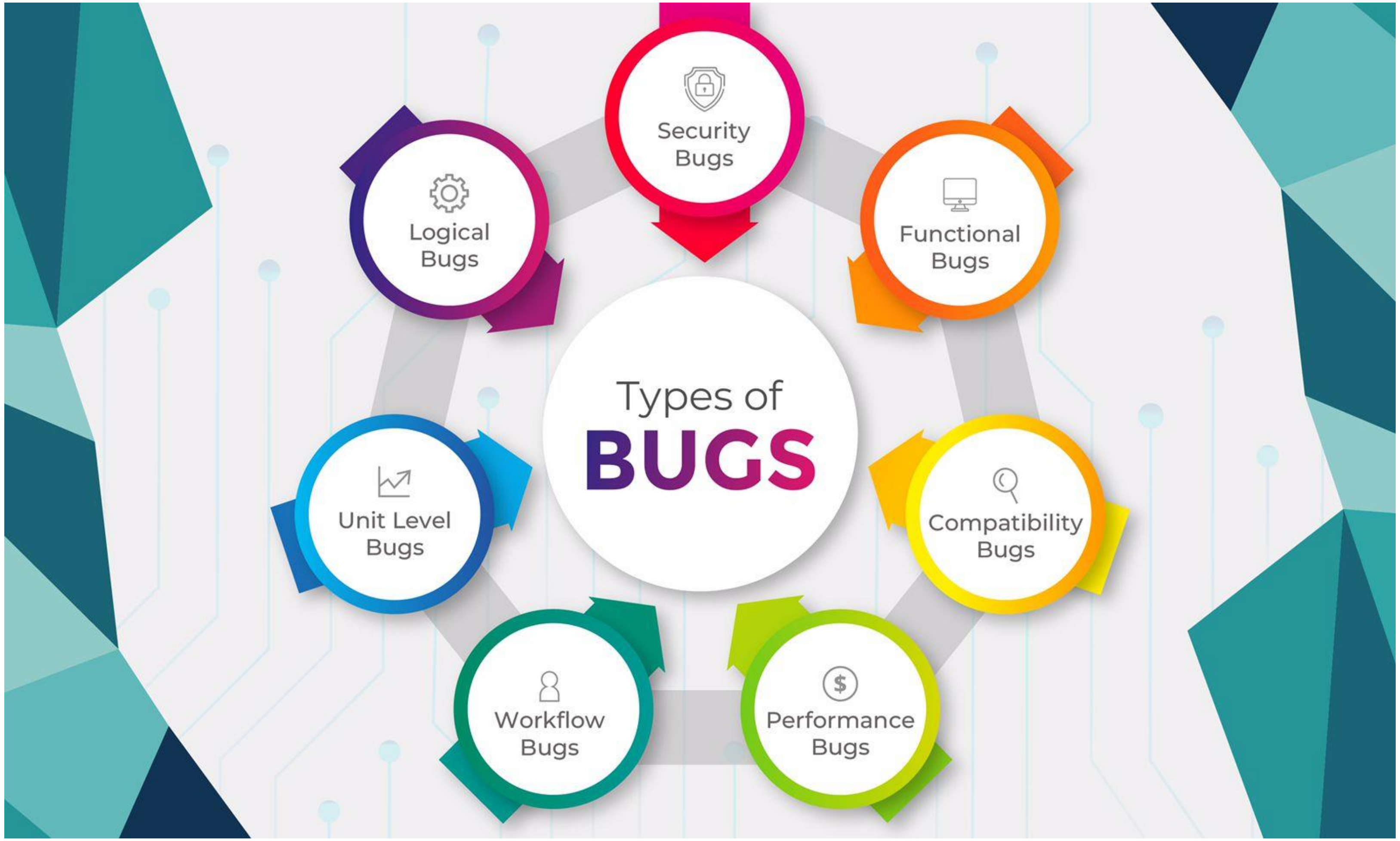
6. LLVM as backend for Postgress SQL (GitHub project)

7. GCC – Optimal code generation for RISC-V architecture

Software Analysis Technology



Software Analysis Technology



Software Analysis Technology

1. Source code clone detection
    - Vulnerable third-party detection
    - Copyright violation
    - Old/buggy code detection
    - Copy-paste errors detection
    - Patch analysis

2. Binary code clone detection
    - Vulnerable third-party detection
    - Libraries identification
    - Old/buggy code detection
    - Versions change analysis
    - Source to binary matching
    - Debug information recovery

Software Analysis Technology

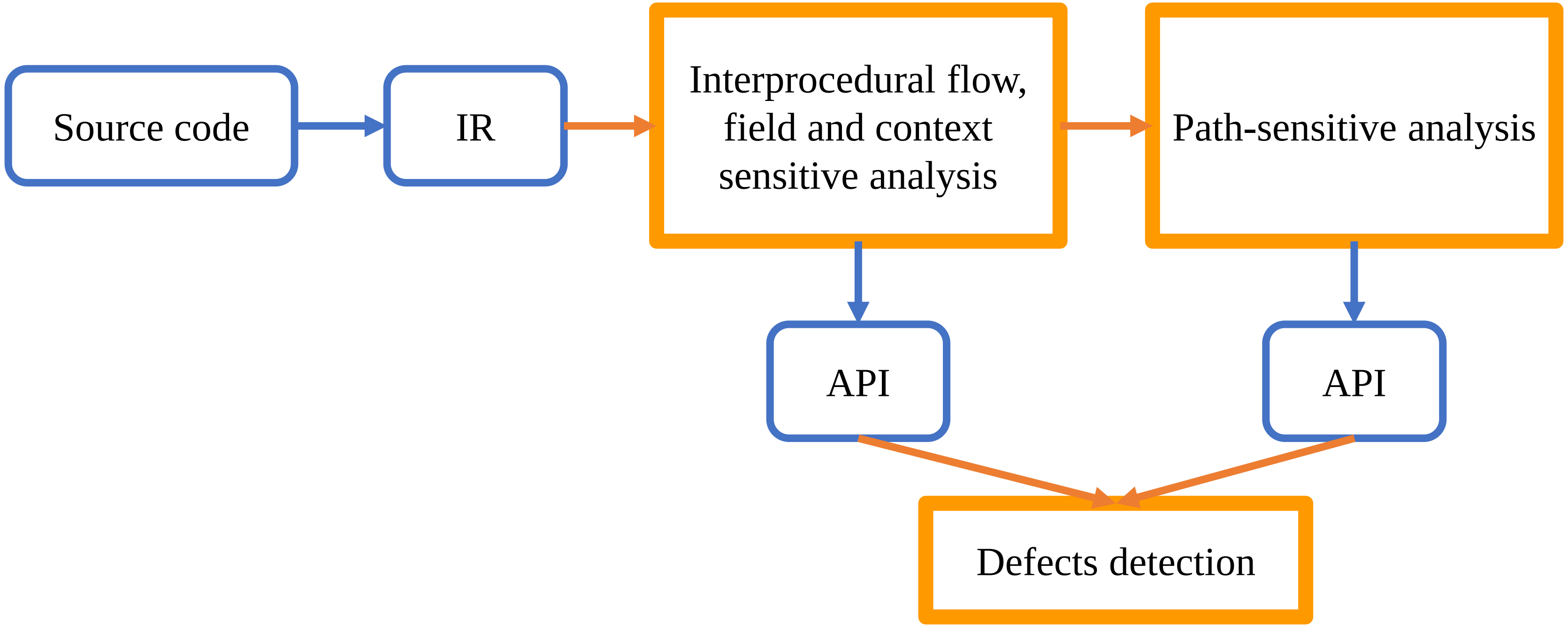
3. Code static analysis
    - Memory leaks detection
    - Buffer overflows detection
    - Etc.

4. Code dynamic analysis
    - Fuzzing
    - Symbolic execution

5. Code analysis framework
    - Code query
    - Mixed analysis with several technologies

Memory Related Errors Detection

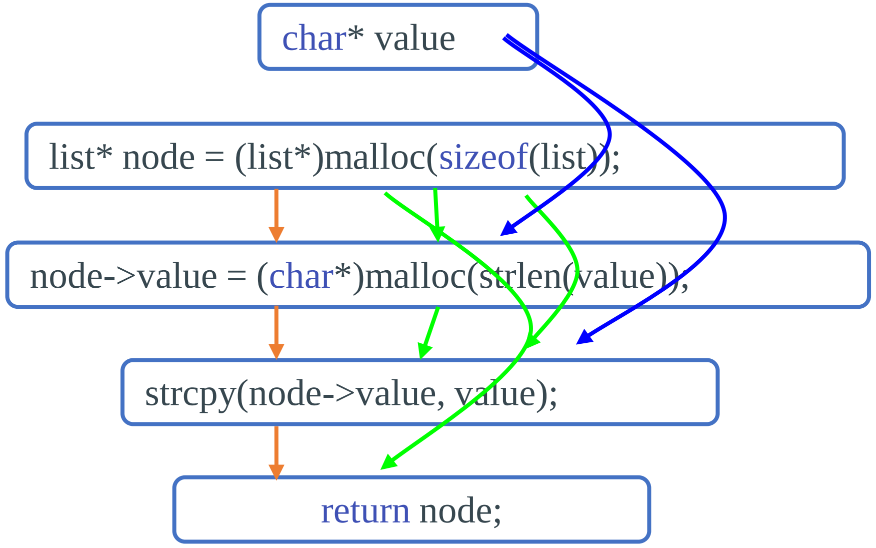
- Memory leak
- Use after free
- Double free



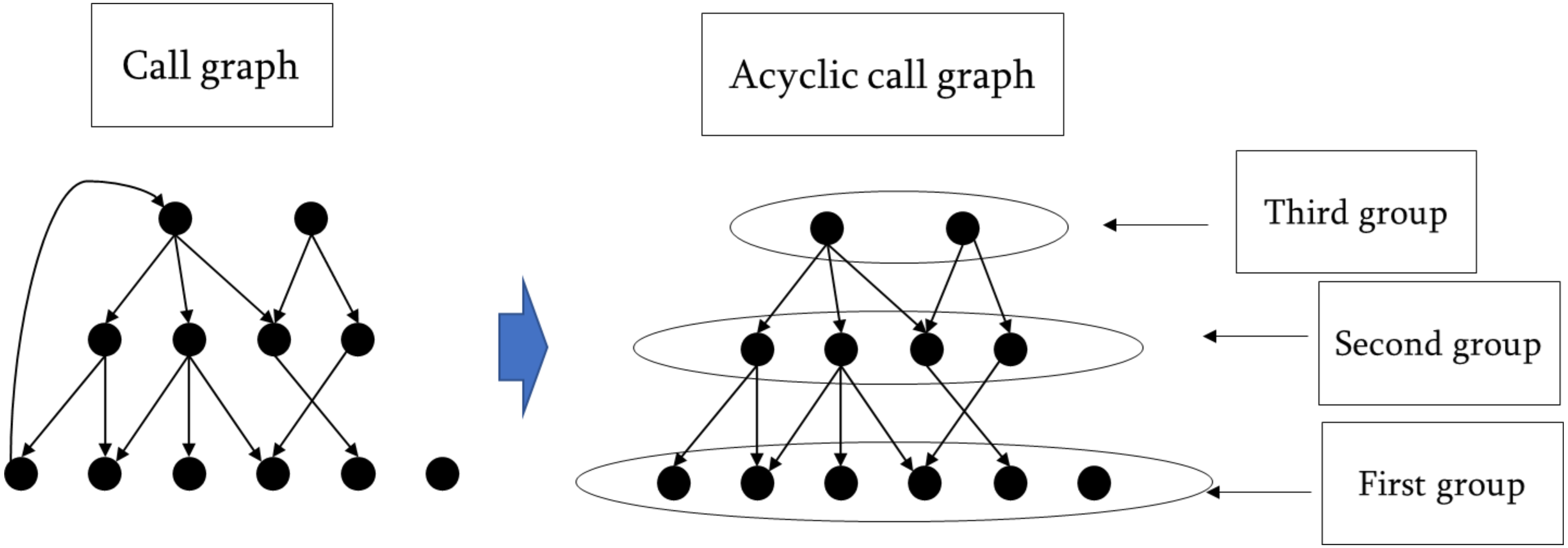
Internal Representation

1. Nodes are IR instructions, function arguments, global variables
2. Edges:
   - Data dependence between two instructions
   - Control flow between two instructions
   - Function argument to user instruction
   - Global variable to user instruction

```
list* allocateNode(char* value) {
   list* node = (list*)malloc(sizeof(list));
   node->value = (char*)malloc(strlen(value));
   strcpy(node->value, value);
   return node;
}
```


char* value
list* node = (list*)malloc(sizeof(list));
node->value = (char*)malloc(strlen(value));
strcpy(node->value, value);
return node;

Interprocedural Analysis



Call graph

Acyclic call graph

Third group

Second group

First group

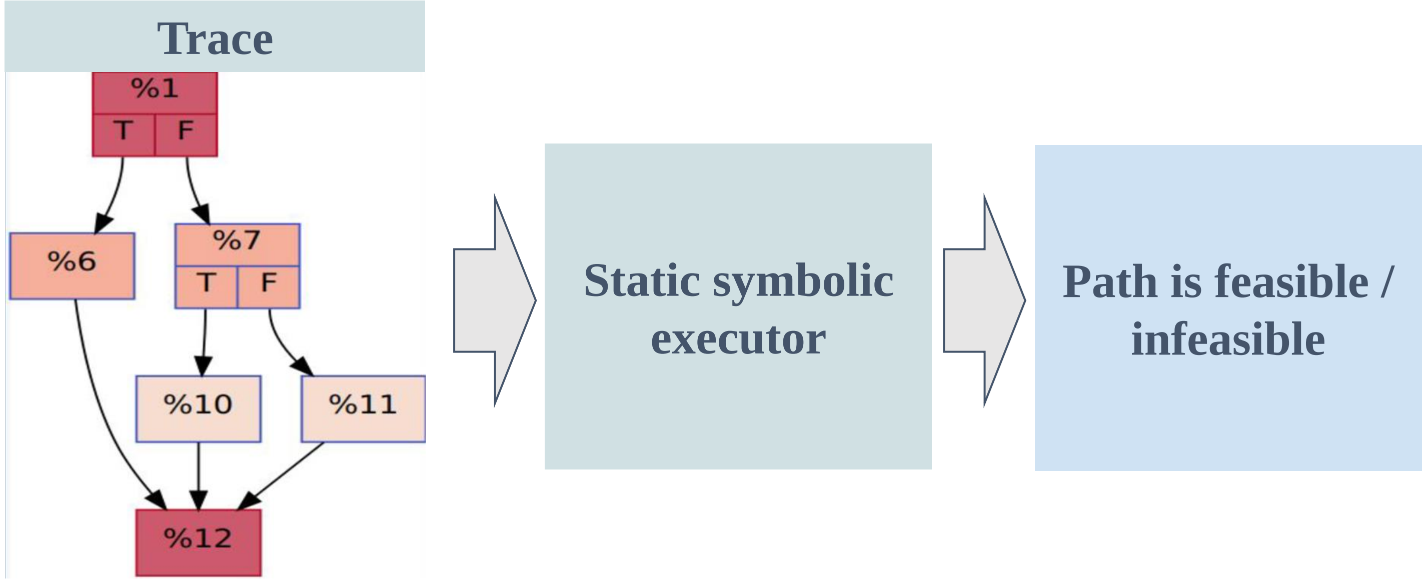
Library/third party functions summaries

- **ALLOC_RETURN,**
- **INNER_ALLOC_RETURNED,**
- **DE_ALLOC_ARGUMENT,**
- **INNER_DE_ALLOC_ARGUMENT,**
- **ALLOC_IN_GLOBAL,**
- **DE_ALLOC_GLOBAL,**
- **ARGUMENT_COPY_TO_ARGUMENT,**
- **ARGUMENT_COPY_TO_GLOBAL,**
- **GLOBAL_RETURNED**
- **…..**

We provide opportunity to **manually add functions' summaries**

Path-sensitive analysis

- Trace is a **subset of functions' basic blocks**, and also contains **important basic blocks**
- Symbolic executor **must execute important basic blocks**
- Symbolic executor **mustn't execute other basic blocks not in the trace**



Trace

Static symbolic executor

Path is feasible / infeasible

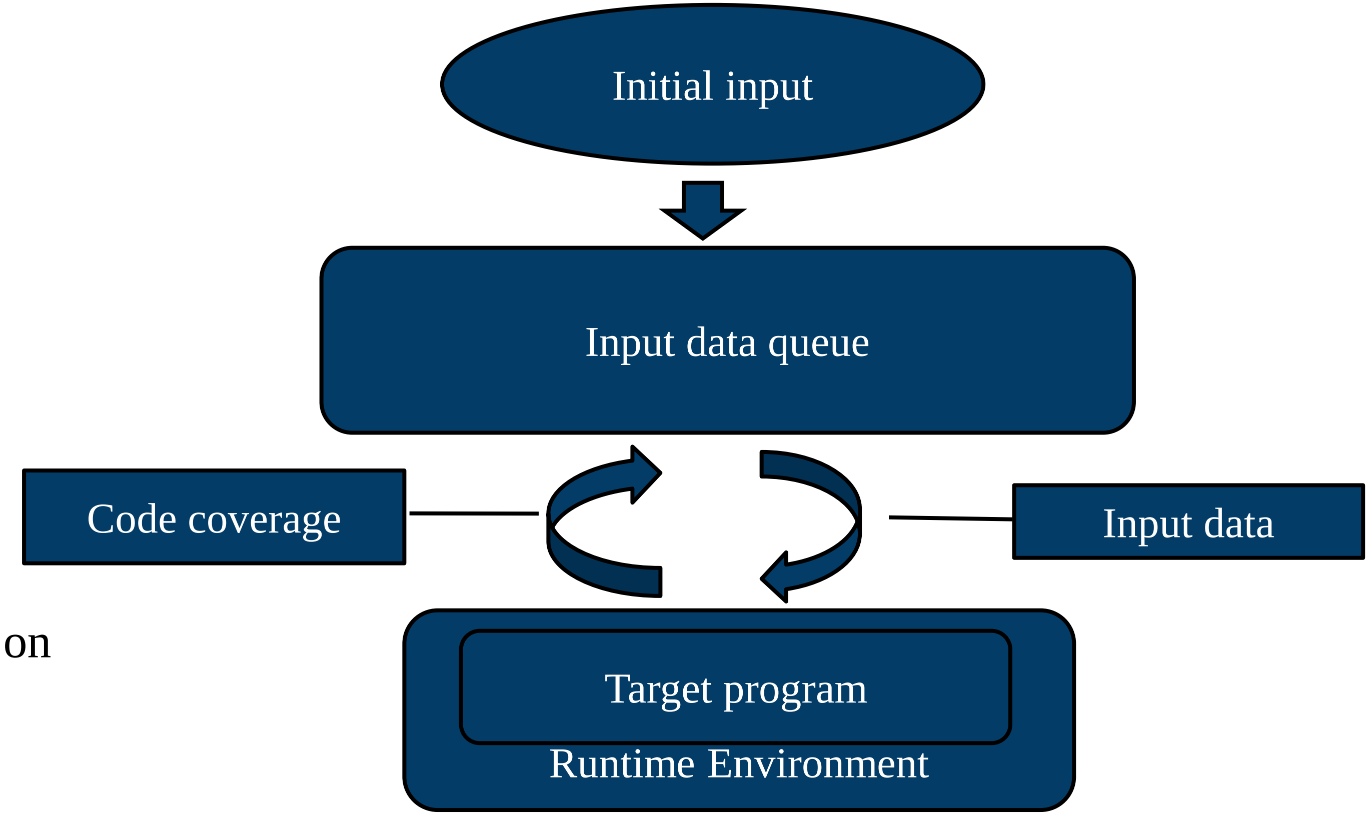
Results on Opensource Projects

- **Confirmed :**
- https://github.com/openssl/openssl/issues/20870
- https://github.com/radareorg/radare2/issues/21705
- https://github.com/radareorg/radare2/issues/21705
- https://github.com/radareorg/radare2/issues/21703
- https://github.com/tmux/tmux/issues/3554
- https://trac.ffmpeg.org/ticket/10342#comment:2

- Found about **229 defects** in Top 100 C projects in github and OSS-fuzz projects, not verified yet

Code Dynamic Analysis: Fuzzing

1. Dumb fuzzing
2. Smart fuzzing
3. Fuzzing can’t generate structured data



Fuzzing tool example: AFL is smart fuzzing tool based on genetic algorithm.

Code Dynamic Analysis: Fuzzing

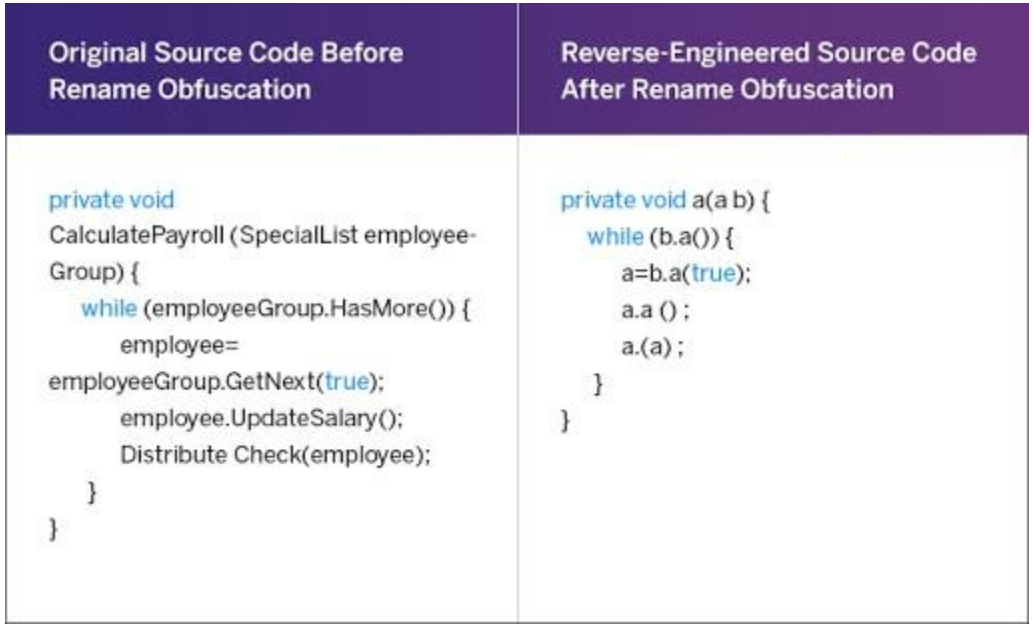
1. BNF grammar fuzzing
2. Directed fuzzing
3. API calls (calls chain) fuzzing
4. Network fuzzing
5. Hybrid fuzzing (DSE + SA + Fuzzing)
6. Fuzzing improvement based on extracted constant values (**Huawei**)

CAST

Compiler Technologies: Obfuscation

90% of security breaches happen because of vulnerabilities in the code.

Source: *Department of Homeland Security*



Original Source Code Before Rename Obfuscation

```
private void
CalculatePayroll (SpecialList employee-
Group) {
    while (employeeGroup.HasMore()) {
         employee=
employeeGroup.GetNext(true);
         employee.UpdateSalary();
         Distribute Check(employee);
     }
}
```

Reverse-Engineered Source Code After Rename Obfuscation

```
private void a(a b) {
   while (b.a()) {
        a=b.a(true);
        a.a () ;
        a.(a) ;
    }
}
```

Compiler Technologies: Obfuscation

Code obfuscation can be used for:

1. Software security improvements
2. Protection from reverse engineering

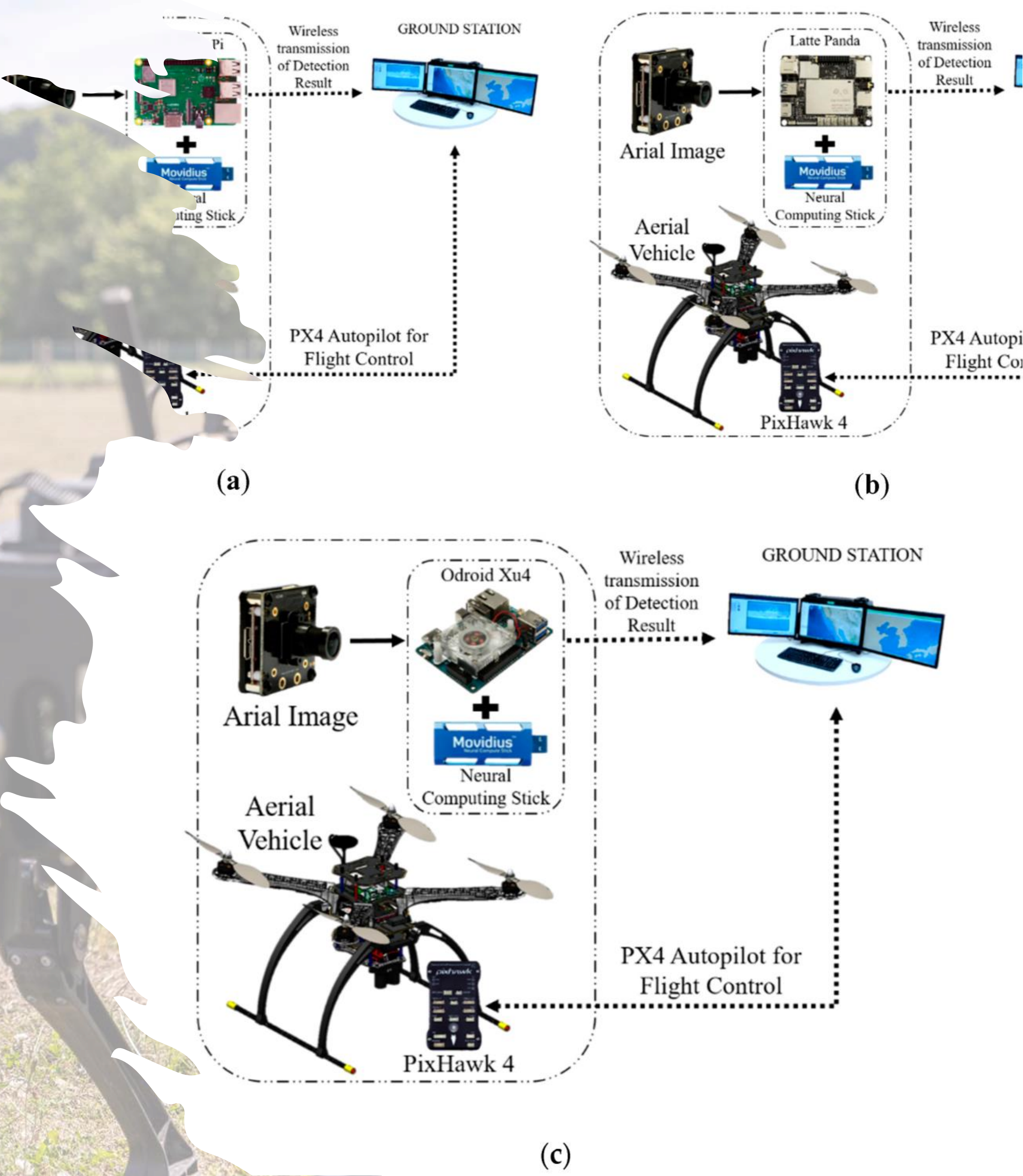
LLVM based source code obfuscator (data and control flaw obfuscation):

1. Functions merging
2. Local variables reordering on stack
3. Addition of redundant calculation
4. Addition of branching instruction
5. Addition of extra functions call

Metrics to calculate possible slow down for each type of change.

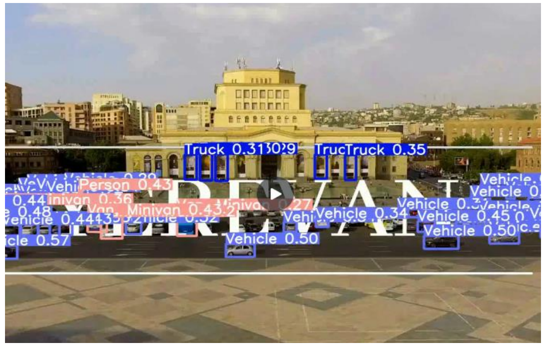
CAST

Autonomous Systems



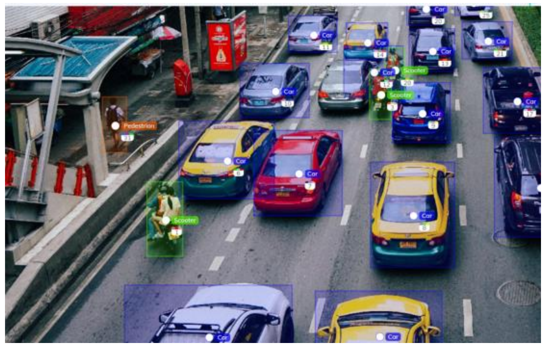
Object Detection

Object detection from drones involves the use of advanced computer vision techniques to identify and locate specific objects or targets within the captured aerial imagery, enabling applications such as surveillance, search and rescue, and environmental monitoring.



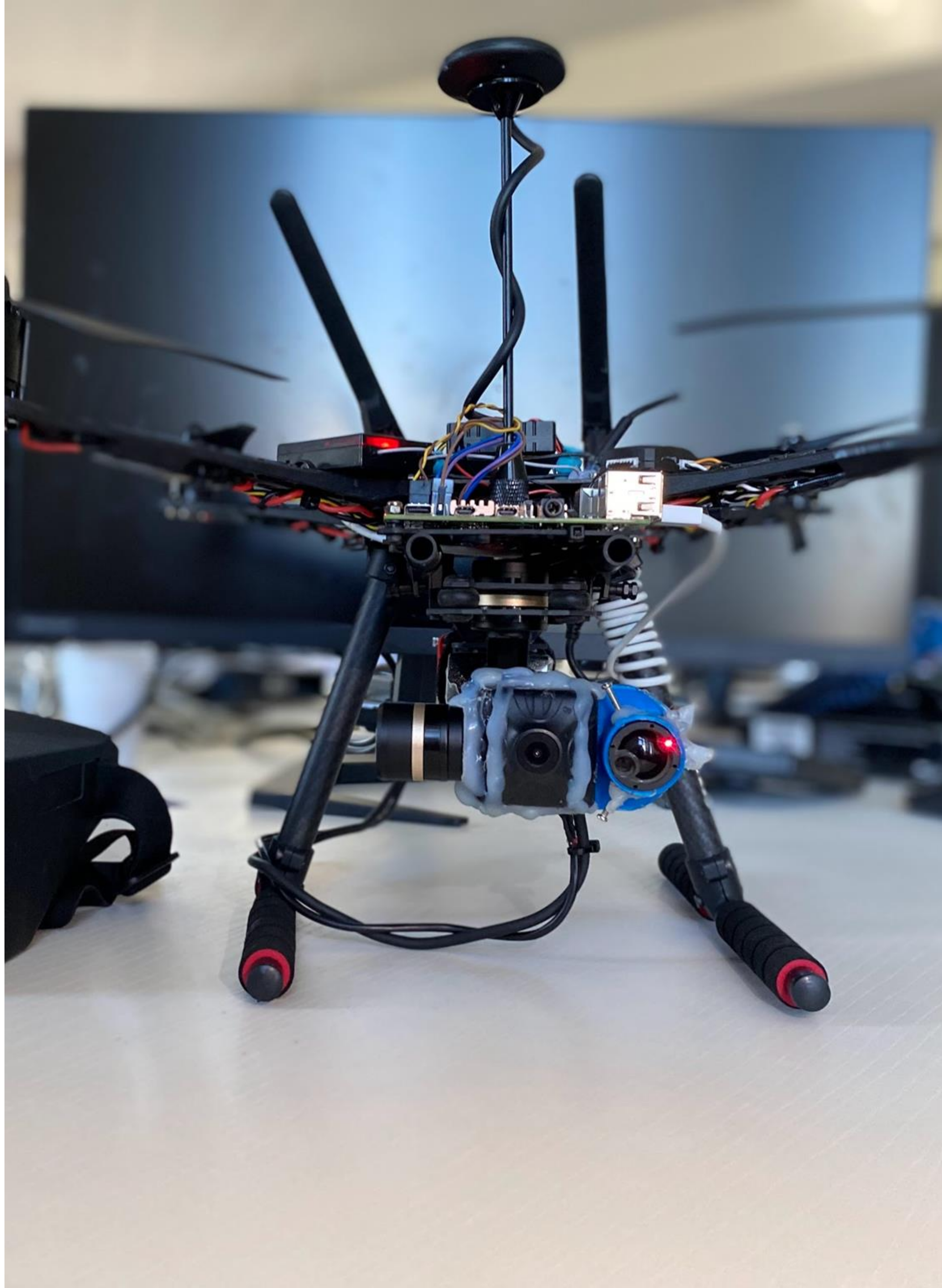
Object Tracking

Object tracking from drones employs sophisticated algorithms to continuously monitor and follow specific targets within the captured aerial footage, ensuring real-time updates on their movement and location. This capability is instrumental in tasks like tracking vehicles during traffic analysis, and enhancing the efficiency of search operations.



Coordinate Extraction

Project uses a drone equipped with a camera, gimbal, rangefinder, telemetry, GPS, and IMU modules. The drone's camera sends real-time video to a ground station, where an operator can select a specific object of interest. Once the operator selects the object, the drone extract its coordinates.

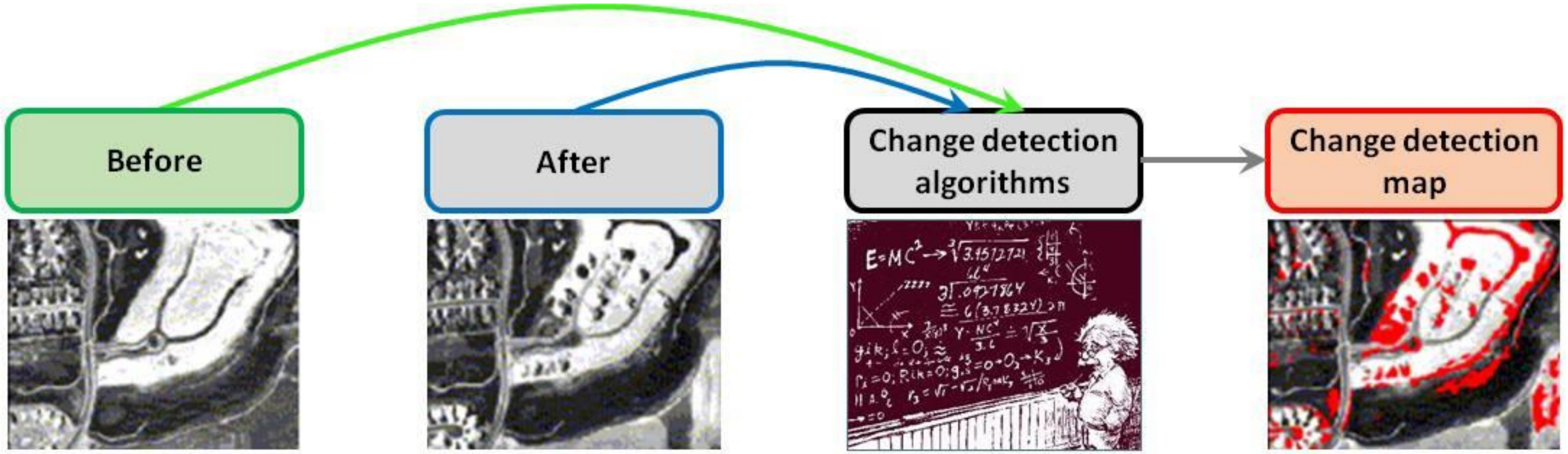


Stitching

The goal of this project is to create an image stitching algorithm capable of seamlessly combining extensive collections of aerial images captured by a swarm of drones. This process aims to generate a comprehensive 2D map that accurately represents the monitored area.

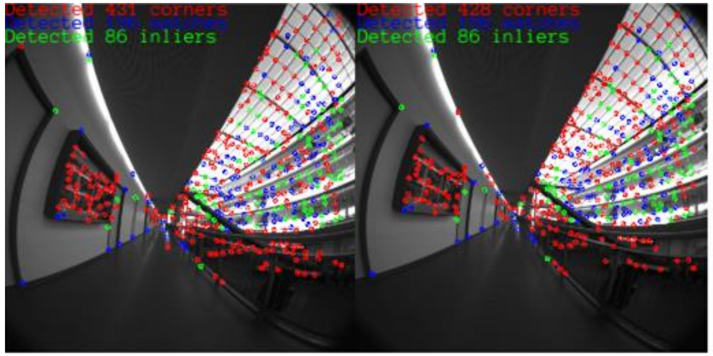
Change Detection

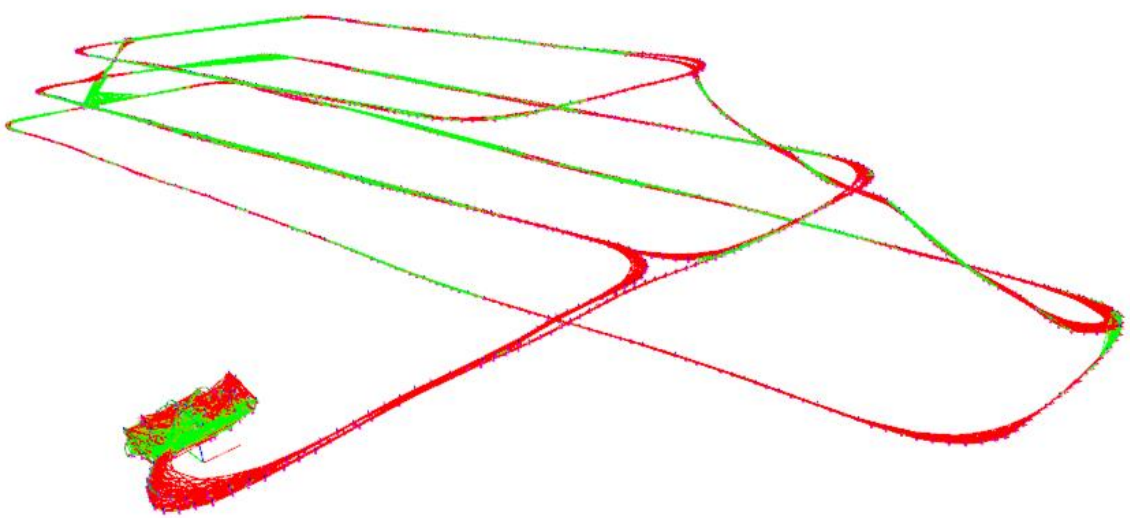
Change Detection aims to identify changes that occur in an image pair taken different times in the same region. Different Deep Learning methods have been considered and tested one of which was proposed by the Change Detection team. The rest are state-of the art methods in the field with some modifications. Future work includes data collection and testing more algorithms available.



Visual Navigation

This project aims real time Visual Inertial Navigation on Robot Operating System 2, synchronizing image frames with Inertial Measurement Unit data.



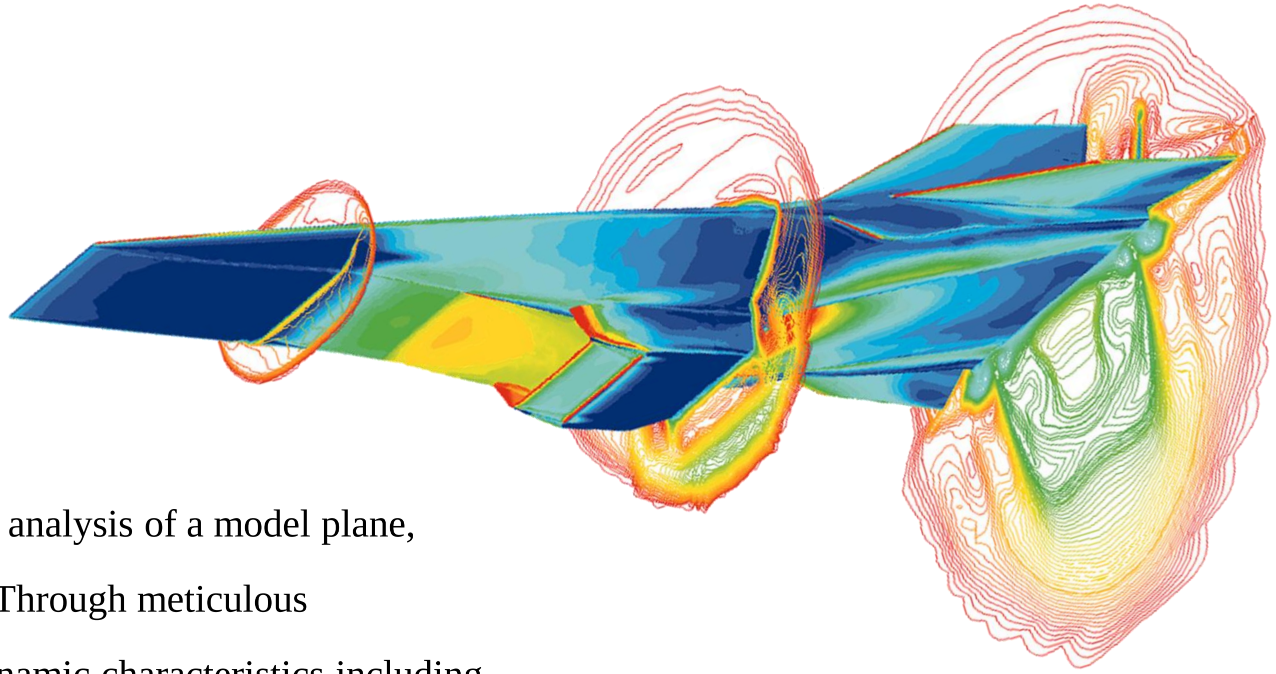


Payload at Wind

The initial goal of this project is to compensate drag forces while payload delivery. Its based on drag calculation that involves wind estimation and provides analytical solution to the problem. The project can be scaled to a small desktop application which visualizes active drag forces on payload and possible drop location.

CAST

Aerodynamics

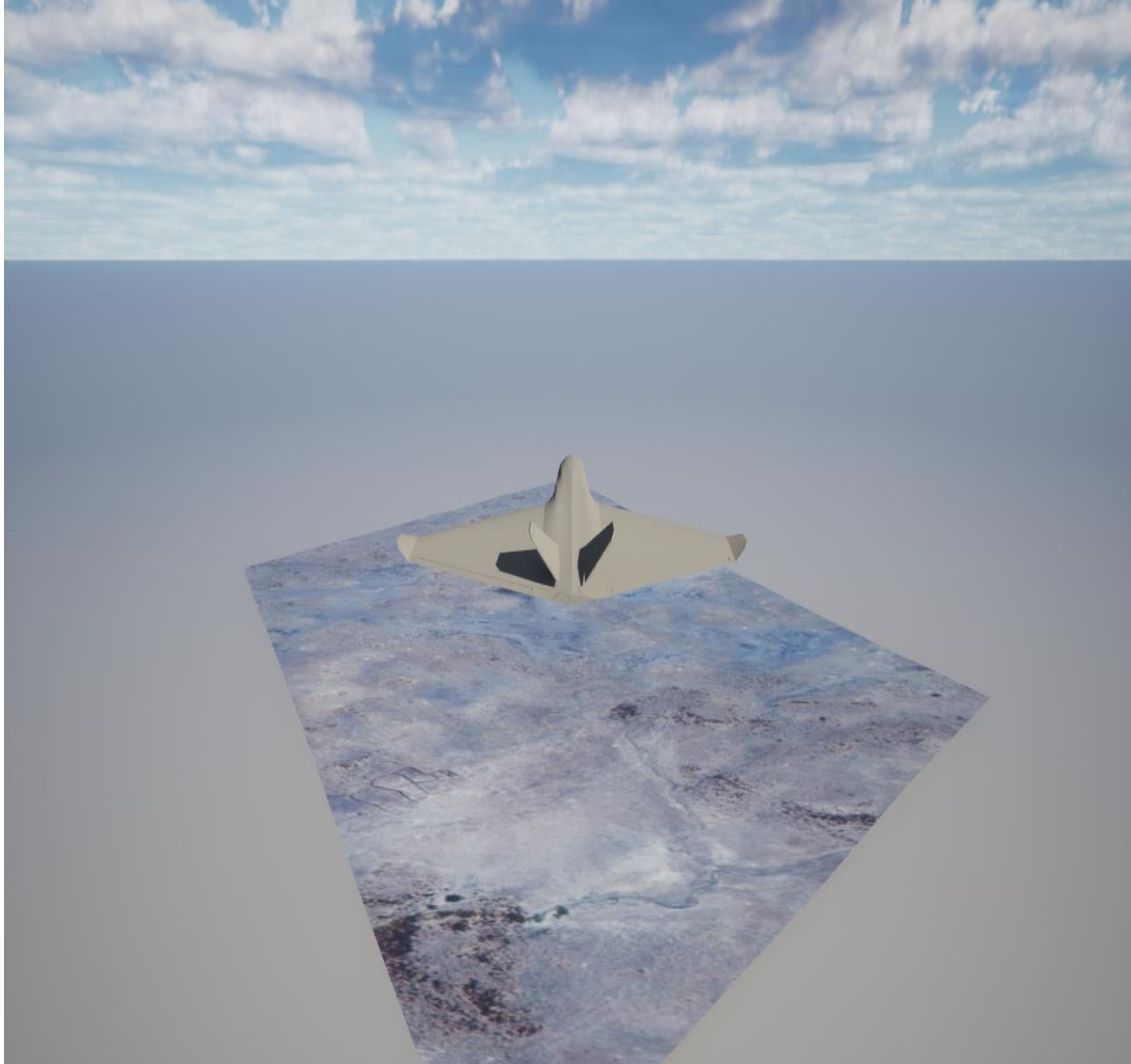


This project aims to conduct a comprehensive aerodynamic analysis of a model plane, focusing on evaluating key flight performance parameters. Through meticulous experimentation and computational simulations, the aerodynamic characteristics including lift, drag, and stability will be thoroughly examined. Development of an environment for modeling the dynamics of flying robotic systems based on aerodynamic analysis.

CAST

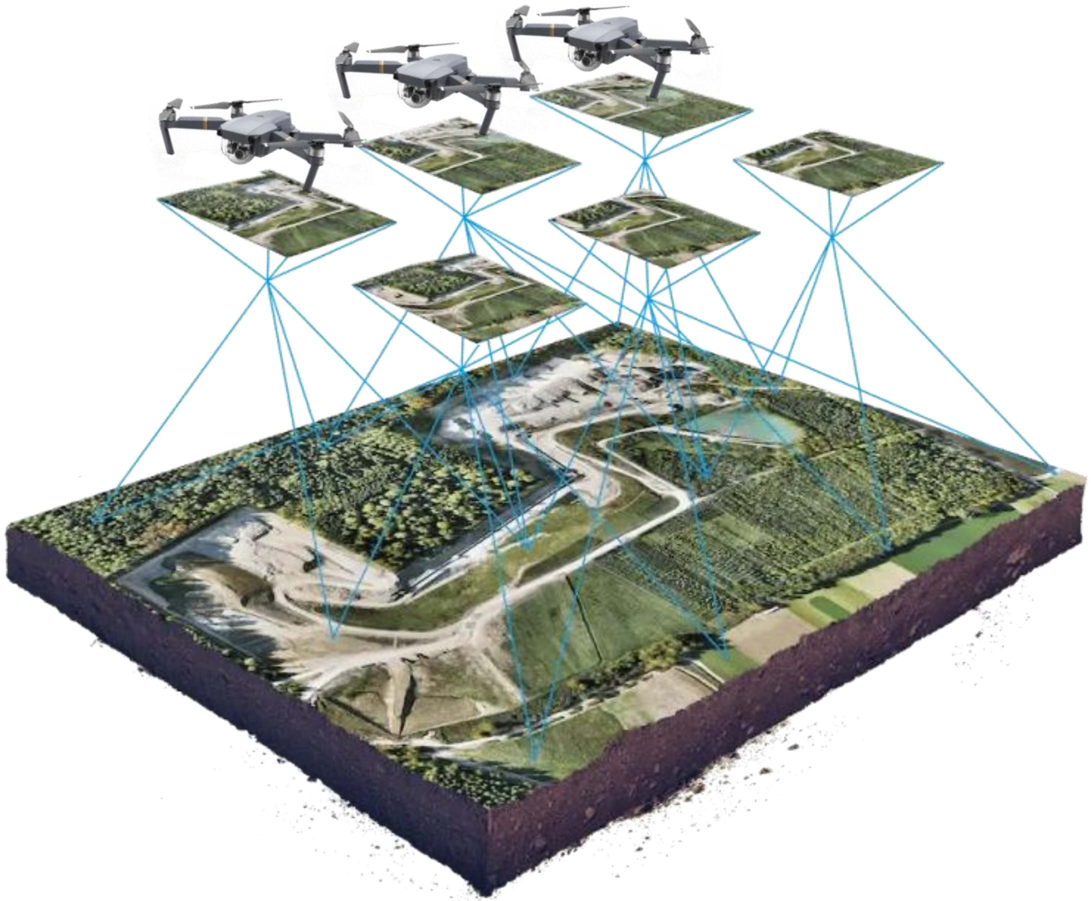
Simulation

Our goal is to develop a flight simulator based on Unreal Engine, which will provide the opportunity to carry out realistic and safe experiments for autonomous flight systems.



Drone Swarm

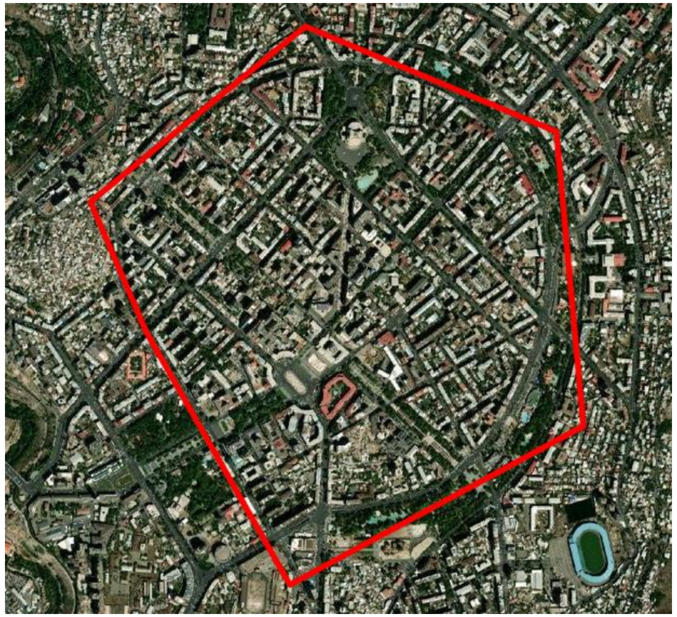
The project aims to devise an efficient algorithm that strategically allocates multiple drones to cover the specified area in the shortest possible time, considering factors like drone speed, area geometry, and obstacle avoidance. By optimizing the distribution and movement of drones, the project seeks to enhance overall coverage effectiveness and minimize operational time.

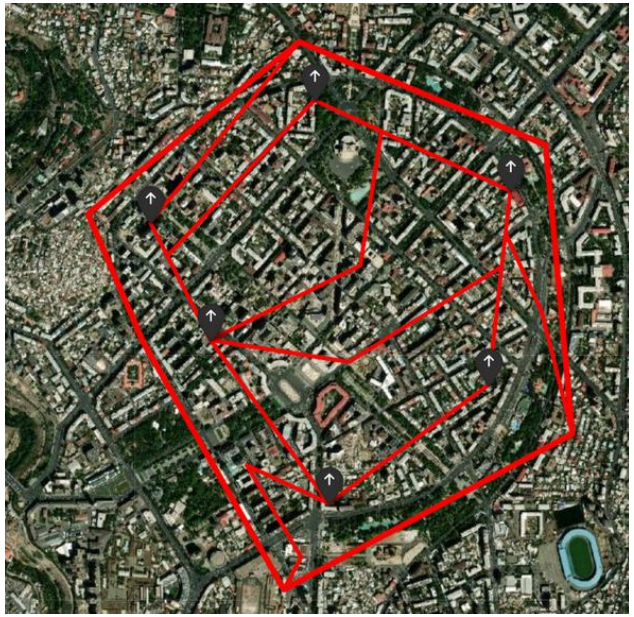


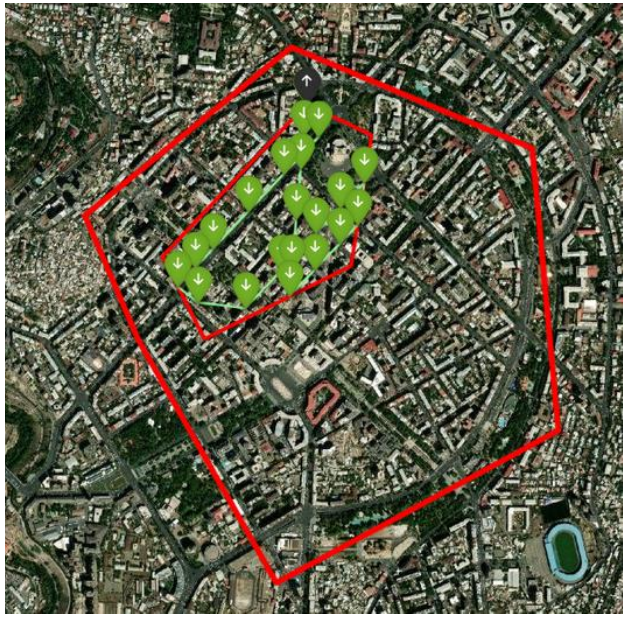
Methodology: Drone Swarm

Solution consists of two parts:

1. Dividing the area into parts.
2. Finding the optimal path for each divided area.

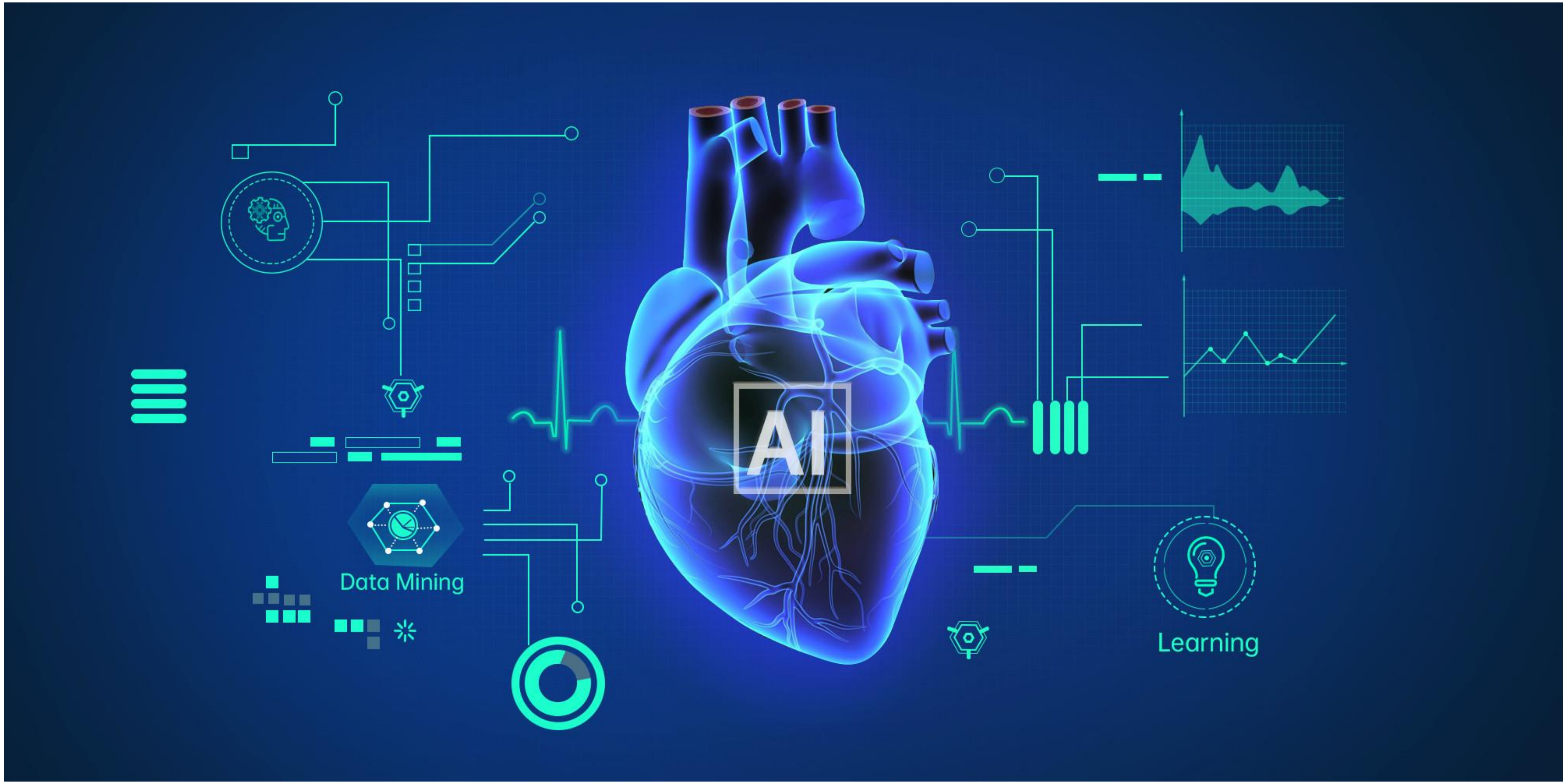




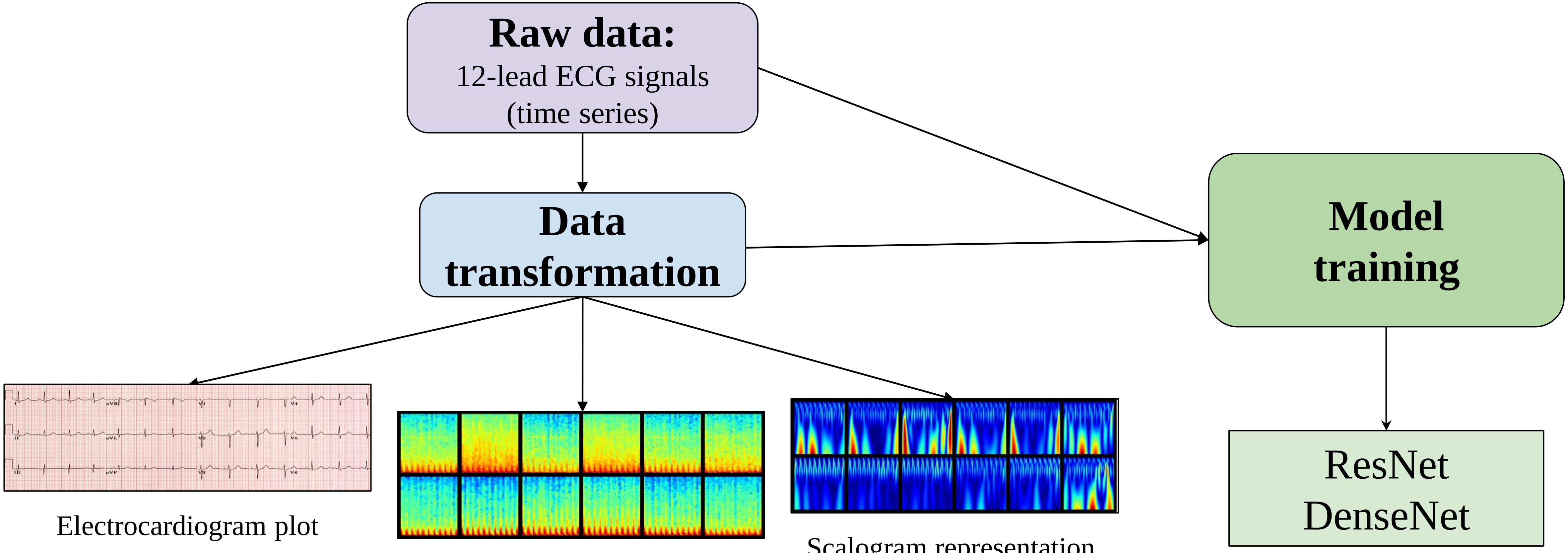


CAST

AI in health



12-lead ECG analysis with data transformation



Raw data:
12-lead ECG signals
(time series)

Data transformation

Model training

ResNet
DenseNet

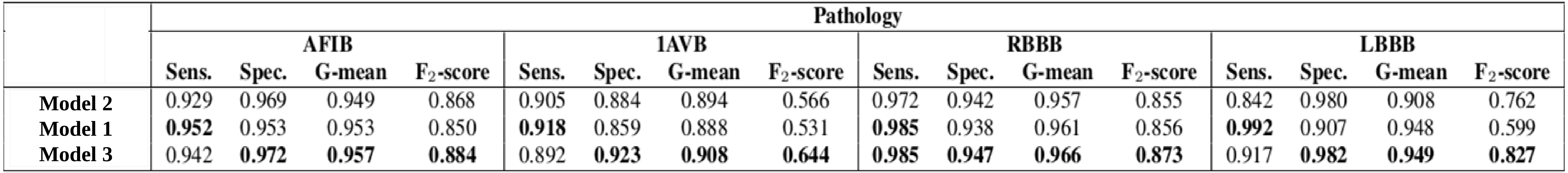
Electrocardiogram plot

Spectrogram representation

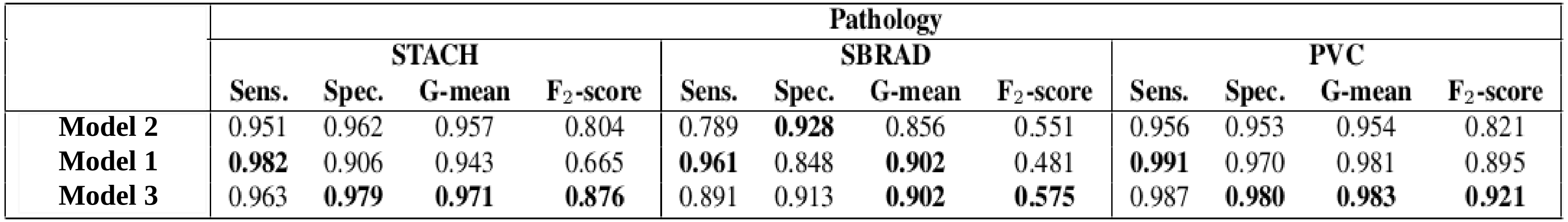
Scalogram representation

12-lead ECG analysis: Results

Comparison of the obtained metrics on PTB-XL dataset



	Pathology															
	AFIB				1AVB				RBBB				LBBB			
	Sens.	Spec.	G-mean	F_2-score	Sens.	Spec.	G-mean	F_2-score	Sens.	Spec.	G-mean	F_2-score	Sens.	Spec.	G-mean	F_2-score
Model 2	0.929	0.969	0.949	0.868	0.905	0.884	0.894	0.566	0.972	0.942	0.957	0.855	0.842	0.980	0.908	0.762
Model 1	**0.952**	0.953	0.953	0.850	**0.918**	0.859	0.888	0.531	**0.985**	0.938	0.961	0.856	**0.992**	0.907	0.948	0.599
Model 3	0.942	**0.972**	**0.957**	**0.884**	0.892	**0.923**	**0.908**	**0.644**	**0.985**	**0.947**	**0.966**	**0.873**	0.917	**0.982**	**0.949**	**0.827**



	Pathology											
	STACH				SBRAD				PVC			
	Sens.	Spec.	G-mean	F_2-score	Sens.	Spec.	G-mean	F_2-score	Sens.	Spec.	G-mean	F_2-score
Model 2	0.951	0.962	0.957	0.804	0.789	**0.928**	0.856	0.551	0.956	0.953	0.954	0.821
Model 1	**0.982**	0.906	0.943	0.665	**0.961**	0.848	**0.902**	0.481	**0.991**	0.970	0.981	0.895
Model 3	0.963	**0.979**	**0.971**	**0.876**	0.891	0.913	**0.902**	**0.575**	0.987	**0.980**	**0.983**	**0.921**

AFIB: Atrial fibrillation

1AVB: First-degree atrioventricular block

RBBB: Right bundle branch block

LBBB: Left bundle branch block

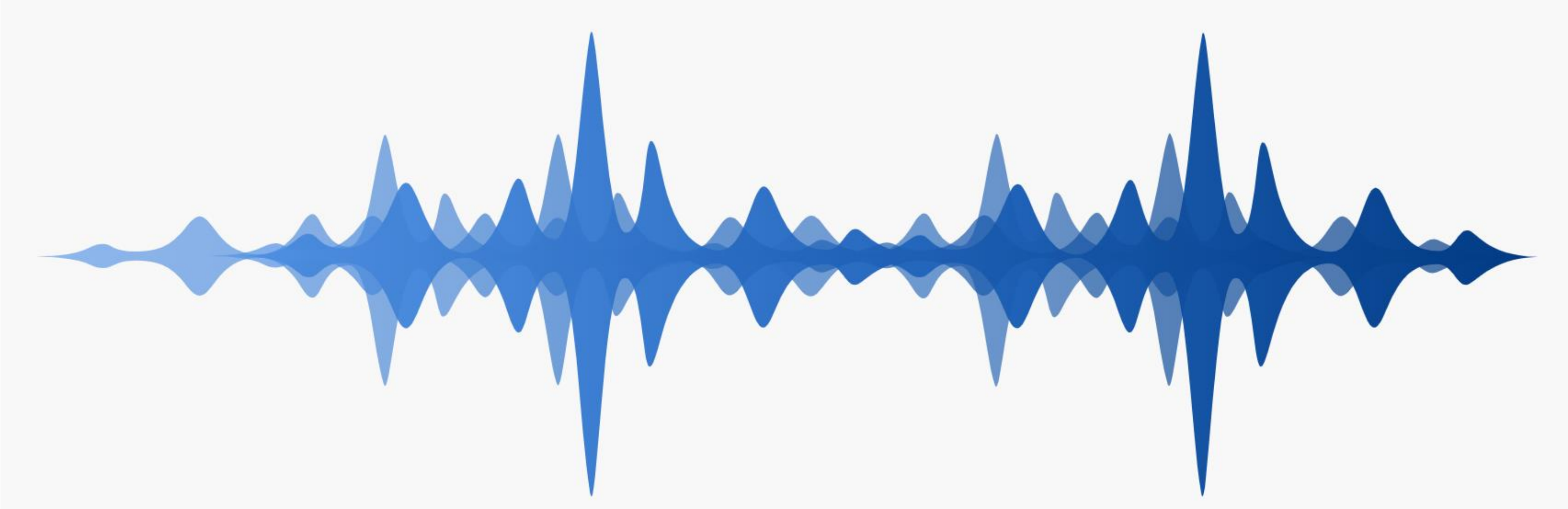
STACH: Sinus tachycardia

SBRAD: Sinus bradycardia

PVC: Premature ventricular contraction

AI based speech technologies

Demo https://wav.am/



RUSSIAN–ARMENIAN
UNIVERSITY

Thank You!