This is an eland

OpenCV

Dependency Extraction

eLand

Eric Rodrigues, Laura Marin, Alp Baran Sirek, Negar Khalilazar, Danny Le

Overview

1. Program Dependency Extraction
2. Implemented Techniques
3. Quantitative and Qualitative Comparison
4. Limitations and Lessons Learned

Program Dependency Extraction

Understand Include srcML

Understand

- SciTools' "Understand" software is an IDE, with multiple tools for code analysis:
  - Code navigation
  - Dependency analysis
  - Visualize with graphs
  - Compliance validation
  - Measure code with metrics

Understand™
by SciTools

Understand Extraction Process

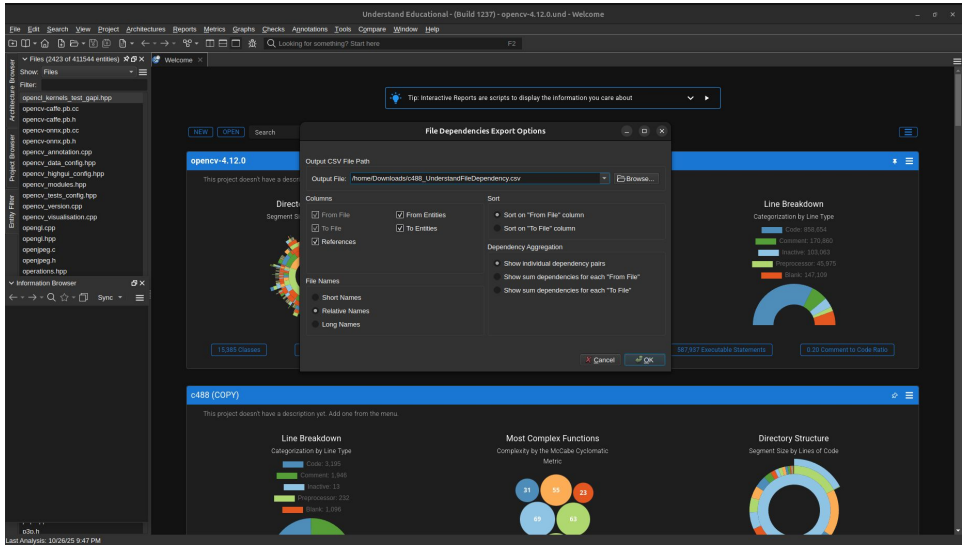
~/opencv-4.12.0/build

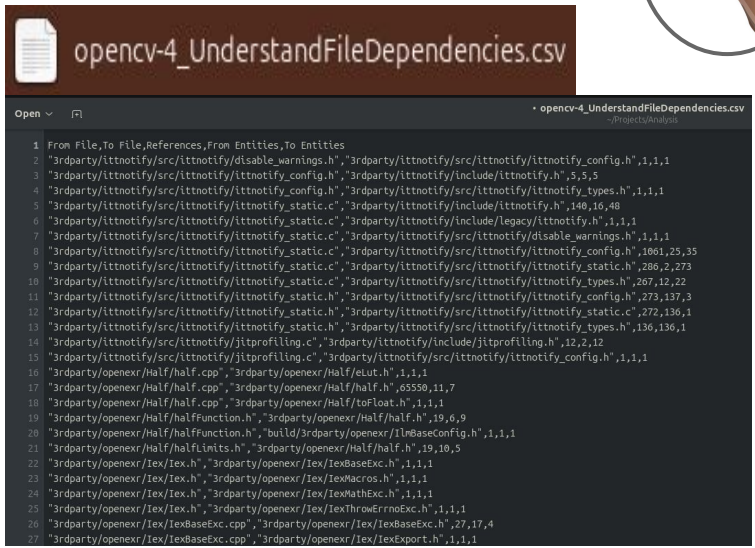
```
:~/opencv-4.12.0/build$ cmake ../build
-- ocv_init_download: OpenCV source tree is not fetched as git repository. 3rdparty resources will be downloaded from github.com by
 default.
-- Detected processor: x86_64
-- Looking for ccache - not found
-- Could NOT find AVIF (missing: AVIF_LIBRARY AVIF_INCLUDE_DIR)
-- Could NOT find OpenJPEG (minimal suitable version: 2.0, recommended version >= 2.3.1). OpenJPEG will be built from sources
-- OpenJPEG: VERSION = 2.5.3, BUILD = opencv-4.12.0-openjp2-2.5.3
-- OpenJPEG libraries will be built from sources: libopenjp2 (version "2.5.3")
-- libva: missing va.h header (VA_INCLUDE_DIR)
-- found Intel IPP (ICV version): 2022.1.0 [2022.1.0]
-- at: /home/metalgeardull/opencv-4.12.0/build/3rdparty/ippicv/ippicv_lnx/icv
-- found Intel IPP Integration Wrappers sources: 2022.1.0
-- at: /home/metalgeardull/opencv-4.12.0/build/3rdparty/ippicv/ippicv_lnx/iw
-- Could NOT find Atlas (missing: Atlas_CBLAS_INCLUDE_DIR Atlas_CLAPACK_INCLUDE_DIR Atlas_CBLAS_LIBRARY Atlas_BLAS_LIBRARY Atlas_LA
PACK_LIBRARY)
-- Could NOT find BLAS (missing: BLAS_LIBRARIES)
-- Could NOT find LAPACK (missing: LAPACK_LIBRARIES)
    Reason given by package: LAPACK could not be found because dependency BLAS could not be found.

-- Could NOT find JNI (missing: JAVA_INCLUDE_PATH JAVA_INCLUDE_PATH2 AWT)
-- VTK is not found. Please set -DVTK_DIR in CMake to VTK build directory, or to VTK install subdirectory with VTKConfig.cmake file
-- Checking for module 'gtk+-2.0'
--   Package 'gtk+-2.0', required by 'virtual:world', not found
-- Checking for modules 'libavcodec;libavformat;libavutil;libswscale'
--   Package 'libavcodec', required by 'virtual:world', not found
--   Package 'libavformat', required by 'virtual:world', not found
--   Package 'libavutil', required by 'virtual:world', not found
--   Package 'libswscale', required by 'virtual:world', not found
-- FFMPEG is disabled. Required libraries: libavcodec;libavformat;libavutil;libswscale. Missing libraries: libavcodec;libavformat;l
ibavutil;libswscale
-- Checking for module 'gstreamer-base-1.0'
```

compile_commands.json

- Configure OpenCV’s build system to create “compile_commands.json”.

Understand Extraction Process





- SciTools' "Understand" application analyses the file dependencies, uses the .json file for accuracy, and exports the file dependencies a .csv file

Understand Extraction Process

```
# EECS 4314 Lab Script
#       This script is used to transform the understand csv data into TA format
#
#/usr/bin/perl
#!"C:\Strawberry\perl\bin\perl.exe"

use strict;
use warnings;

my ($header, $from, $to, $reference, $v1, $v2, $output_file);
my (%FILE_HASH, %CALL_HASH, $name, $value, $count, $hash, $line);

$header = 0;

open INPUT, "$ARGV[0]";

if ($ARGV[0] =~ /(.*)\.csv/) {
        $output_file = $1 . ".raw.ta";
}

open OUTPUT, ">$output_file";

print OUTPUT "FACT TUPLE :\n";


while (<INPUT>) {
        $line =~;
        chomp $line;

        if ($header == 0) {
                $header++;
                next;
        }

        ($from, $to, $reference, $v1, $v2) = split(/,/, $line);

        # comment out if we don't want to consider header files
#       if ($from =~ /\.h/ or $to =~ /\.h/) {
#               next;
#       }

        $FILE_HASH{$from}++;
        $FILE_HASH{$to}++;
        $CALL_HASH{$from}{$to}++;
}

# output the list of files
while (($name, $value) = (each %FILE_HASH)) {
        # $INSTANCE cocoxen.o[.text+0x0] function
        print OUTPUT "$INSTANCE $name cFile\n";
}

# output the list of call relations
while (($name, $hash) = (each %CALL_HASH)) {
        while (($value, $count) = (each %$hash)) {
                # clinks cocoxen.o[.text+0x173] semantics.o[.bss+0x0]
                # print OUTPUT "contain $name $value\n";
                print OUTPUT "clinks $name $value\n";
        }
}

close INPUT;
close OUTPUT;
```

opencv-4_UnderstandFileDependencies.raw.ta

Open ~/Projects/Analysis

opencv-4_UnderstandFileDependencies.raw.ta

```
FACT TUPLE :
$INSTANCE "3rdparty/include/opencl/1.2/CL/cl.h" cFile
$INSTANCE "3rdparty/ittnotify/include/ittnotify.h" cFile
$INSTANCE "3rdparty/ittnotify/include/jitprofiling.h" cFile
$INSTANCE "3rdparty/ittnotify/include/legacy/ittnotify.h" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/disable_warnings.h" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/ittnotify_config.h" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/ittnotify_static.c" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/ittnotify_static.h" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/ittnotify_types.h" cFile
$INSTANCE "3rdparty/ittnotify/src/ittnotify/jitprofiling.c" cFile
$INSTANCE "3rdparty/openexr/Half/eLut.h" cFile
$INSTANCE "3rdparty/openexr/Half/half.cpp" cFile
$INSTANCE "3rdparty/openexr/Half/half.h" cFile
$INSTANCE "3rdparty/openexr/Half/halfFunction.h" cFile
$INSTANCE "3rdparty/openexr/Half/halfLimits.h" cFile
```

- A perl script reads the csv file to create a .ta file

Understand Extraction Process

```java
import java.io.*;
import java.util.*;

public class GenerateContainRelations {
    Run | Debug
    public static void main(String[] args) {
        String inputFile = "opencv-4_UnderstandFileDependencies.raw.ta";
        String outputFile = "opencv-4_UnderstandFileDependencies.contain";

        Set<String> seenRelations = new HashSet<>();

        try (BufferedReader br = new BufferedReader(new FileReader(inputFile));
             PrintWriter pw = new PrintWriter(new FileWriter(outputFile))) {

            String line;
            while ((line = br.readLine()) != null) {
                line = line.trim();

                if (!line.startsWith("$INSTANCE")) continue;

                int start = line.indexOf('"');
                int end = line.lastIndexOf('"');
                if (start == -1 || end == -1 || start >= end) continue;

                String path = line.substring(start + 1, end);
                String[] parts = path.split("/");

                for (int i = 0; i < parts.length - 1; i++) {
                    String parentPath = String.join("/", Arrays.copyOfRange(parts, 0, i + 1));
                    String childPath = String.join("/", Arrays.copyOfRange(parts, 0, i + 2));
                    String relation = "contain \"" + parentPath + "\" \"" + childPath + "\"";

                    if (seenRelations.add(relation)) {
                        pw.println(relation);
                    }
                }
            }

            System.out.println("Containment relationships written to " + outputFile);

        } catch (IOException e) {
            System.err.println("Error processing file: " + e.getMessage());
        }
    }
}
```

opencv-4_UnderstandFileDependencies.contain

Open

opencv-4_UnderstandFileDependencies.contain
~/Projects/ExtractContains

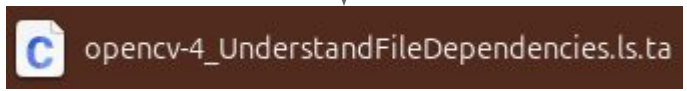
```
contain "3rdparty" "3rdparty/include"
contain "3rdparty/include" "3rdparty/include/opencl"
contain "3rdparty/include/opencl" "3rdparty/include/opencl/1.2"
contain "3rdparty/include/opencl/1.2" "3rdparty/include/opencl/1.2/CL"
contain "3rdparty/include/opencl/1.2/CL" "3rdparty/include/opencl/1.2/CL/cl.h"
contain "3rdparty" "3rdparty/ittnotify"
contain "3rdparty/ittnotify" "3rdparty/ittnotify/include"
contain "3rdparty/ittnotify/include" "3rdparty/ittnotify/include/ittnotify.h"
contain "3rdparty/ittnotify/include" "3rdparty/ittnotify/include/jitprofiling.h"
contain "3rdparty/ittnotify/include" "3rdparty/ittnotify/include/legacy"
contain "3rdparty/ittnotify/include/legacy" "3rdparty/ittnotify/include/legacy/ittnotify.h"
contain "3rdparty/ittnotify" "3rdparty/ittnotify/src"
contain "3rdparty/ittnotify/src" "3rdparty/ittnotify/src/ittnotify"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/disable_warnings.h"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/ittnotify_config.h"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/ittnotify_static.c"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/ittnotify_static.h"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/ittnotify_types.h"
contain "3rdparty/ittnotify/src/ittnotify" "3rdparty/ittnotify/src/ittnotify/jitprofiling.c"
contain "3rdparty" "3rdparty/openexr"
contain "3rdparty/openexr" "3rdparty/openexr/Half"
contain "3rdparty/openexr/Half" "3rdparty/openexr/Half/eLut.h"
contain "3rdparty/openexr/Half" "3rdparty/openexr/Half/half.cpp"
contain "3rdparty/openexr/Half" "3rdparty/openexr/Half/half.h"
contain "3rdparty/openexr/Half" "3rdparty/openexr/Half/halfFunction.h"
contain "3rdparty/openexr/Half" "3rdparty/openexr/Half/halfLimits.h"
```

- Our Java program reads the .ta file to create our containment file.

Understand Extraction Process

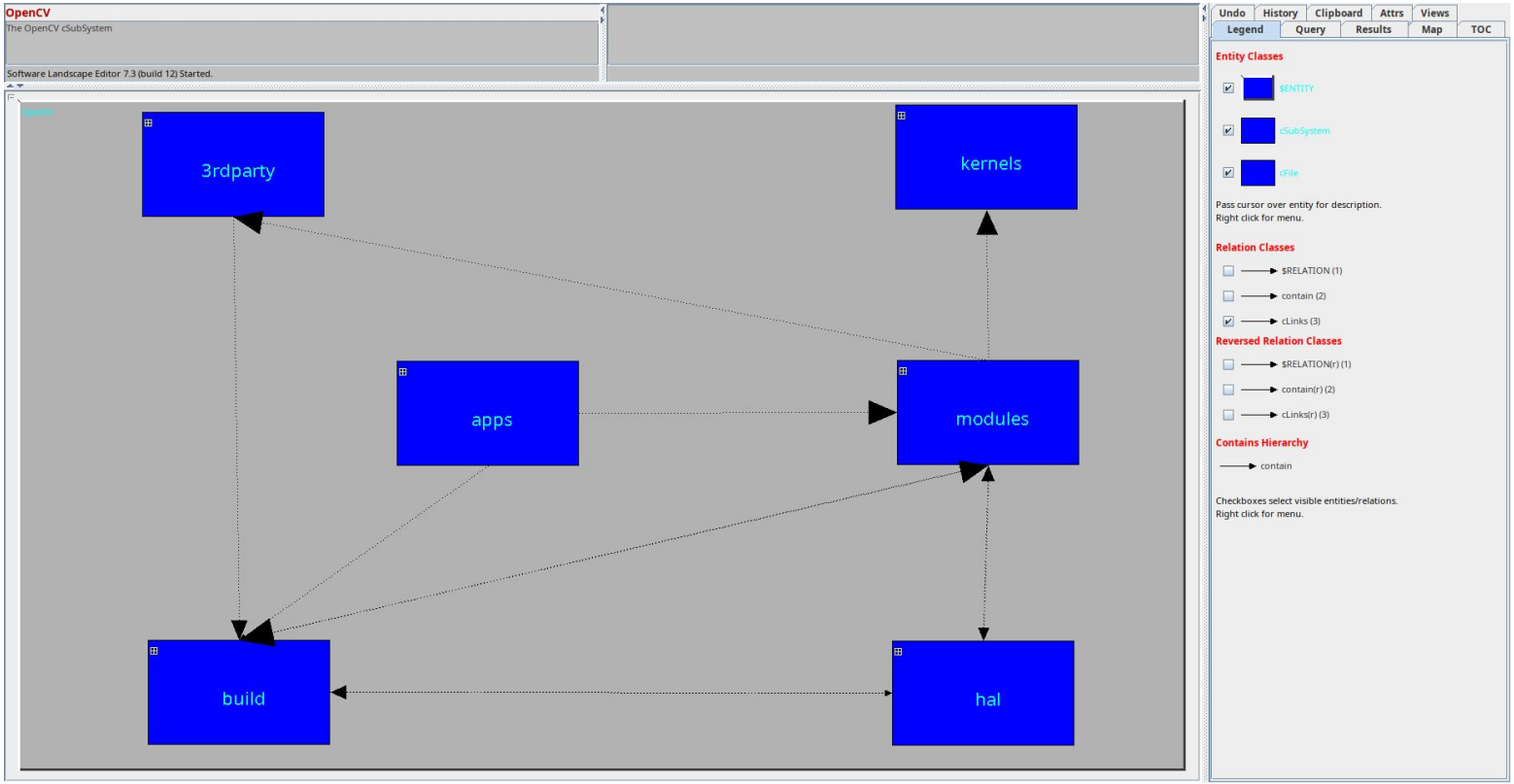
createContainment.sh
~/Projects/Analysis

```
1 #!/bin/bash
2 java -Xms256M -Xmx1024M -classpath ql.jar ca.uwaterloo.cs.ql.Main
  addcontain.ql opencv-4_UnderstandFileDependencies.contain
  opencv-4_UnderstandFileDependencies.raw.ta
  opencv-4_UnderstandFileDependencies.con.ta
3 cat schema.asv.ta opencv-4_UnderstandFileDependencies.con.ta >
  opencv-4_UnderstandFileDependencies.ls.ta
```



- A script reads our containment file to produce a landscape file of OpenCV’s architecture. This is then opened in LSEdit.

Understand Extraction Process



Include Directive Extraction

```
def findFilesFirst(root_path):

    valid_exts = {".c",".cpp",".cxx",".h",".hpp",".hh",".cc",".cpp"}
    for dirpath, _, filenames in os.walk(root_path):
        for filename in filenames:
            if os.path.splitext(filename)[1] in valid_exts:
                yield os.path.join(dirpath, filename)
```

```
regX = re.compile(r'^\s*#\s*include\s*(<|")([^">]+)(>|")')
```

Include Directive Extraction

```
for src_file in files:
    incs = findImports(src_file)

    for inc, line_no in incs:
        found_paths = edtPath(args.root, inc)

        for resolved in found_paths:
            out.write(f"{src_file},{resolved},{line_no}\n")
```

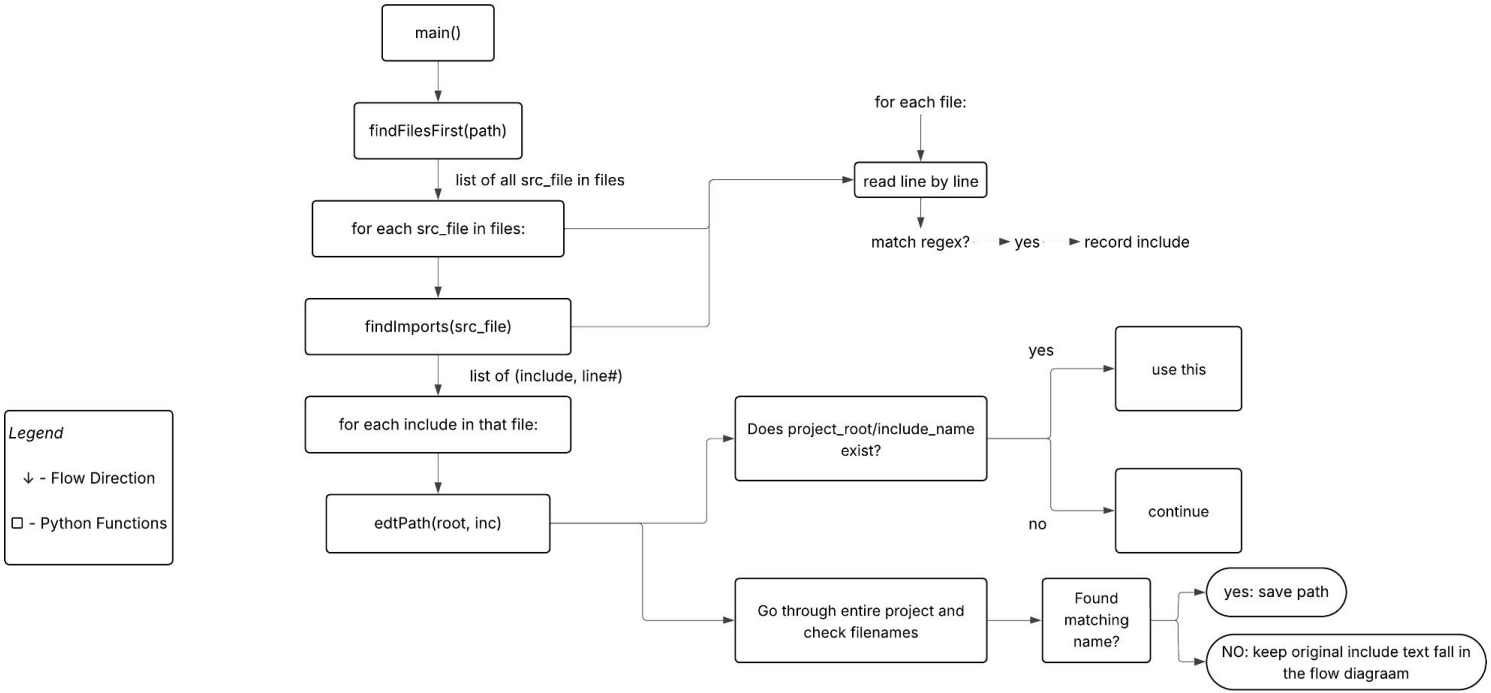
Include Directive Extraction

```python
def findImports(file_path):
    incs = []
    try:
        with open(file_path, "r", errors="ignore") as f:
            for line_number, line in enumerate(f, 1):
                match = regX.match(line)
                if match:
                    inc = match.group(2)
                    incs.append((inc, line_number))
    except Exception:
        pass
    return incs
```

```python
def edtPath(toor, inc):

    results = []
    p1 = os.path.join(toor, inc)
    if os.path.exists(p1):
        results.append(os.path.abspath(p1))

    base = os.path.basename(inc)
    for dirpath, _, filenames in os.walk(toor):
        if base in filenames:
            results.append(os.path.abspath(os.path.join(dirpath, base)))

    if not results:
        return [inc]
    return results
```

Flow Diagram of Include

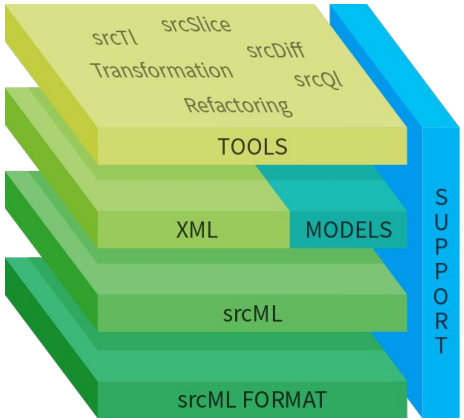
Flow Diagram of include extraction



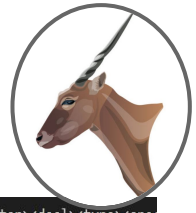
srcML

<srcML>

- Creates a XML representation of source code while keeping originally information intact
- Use XML queries to manipulate data, takes about 3 minutes to ran
- Today we will be using to it extract source code from OpenCV



srcML Extraction Process

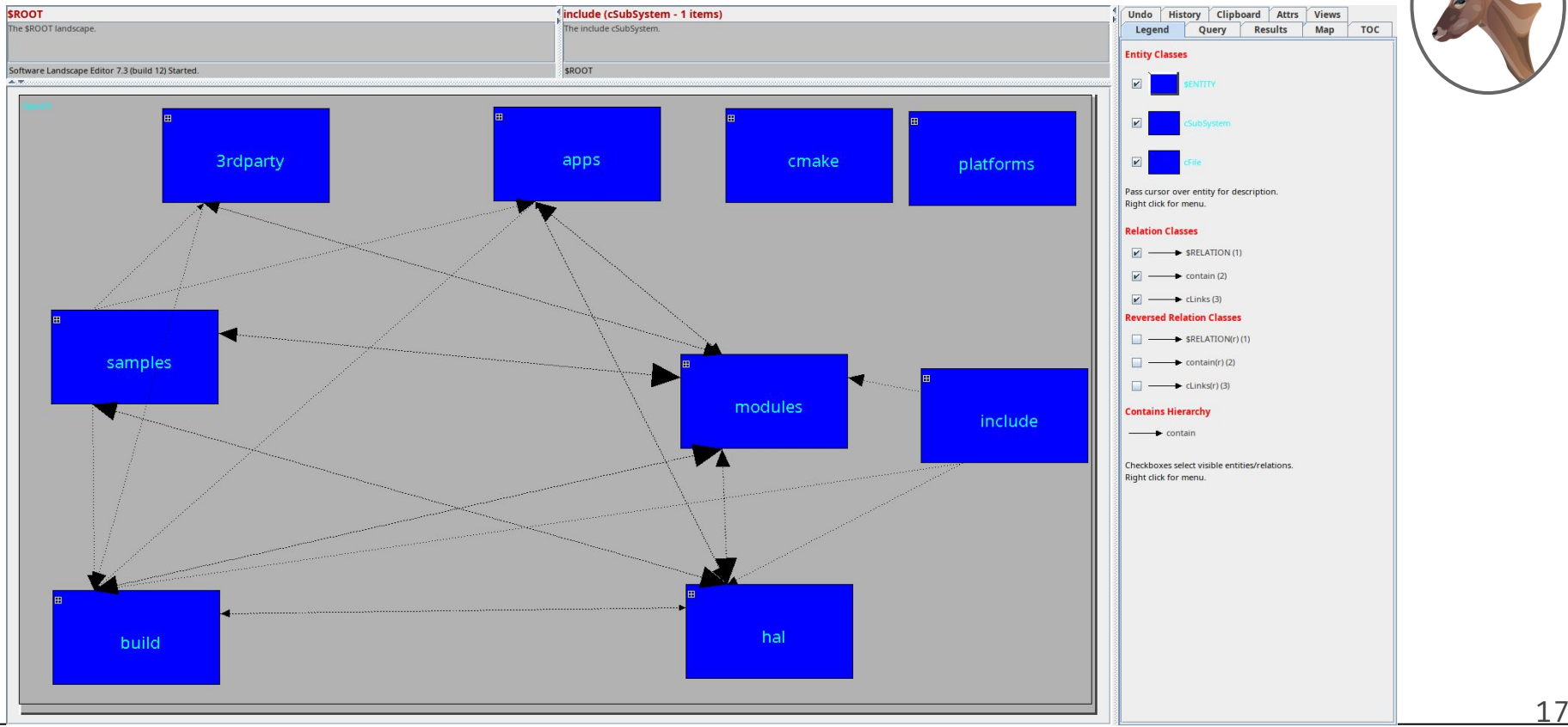


- First a shell script reads every OpenCV file into srcML XML and writes them into "files.txt"
- Then another shell script converts each file into a srcML XML file (one per OpenCV file)
- Finally using a Python script we parse each XML file and extract the dependencies

```
<function><type><name>void</name></type> <name>CopyMatrixToVector</name><parameter_list>(<parameter><decl><type><spec
<block>{<block_content>
    <decl_stmt><decl><type><name>int</name><modifier>*</modifier></type> <name>data</name> <init>= <expr><operator>(
    <for>for <control>(<init><decl><type><name>int</name></type> <name>i</name> <init>= <expr><literal type="number">
    <block>{<block_content>
        <expr_stmt><expr><call><name><name>vector</name><operator>.</operator><name>push_back</name></name><argument_
    </block_content>}</block></for>
</block_content>}</block></function>
</unit>
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<unit xmlns="http://www.srcML.org/srcML/src" xmlns:cpp="http://www.srcML.org/srcML/cpp" revision="1.0.0" language="C+
<comment type="line">// pch.cpp</comment>
<comment type="line">// Include the standard header and generate the precompiled header.</comment>
<comment type="line">//</comment>
```

```
cLinks "build/modules/core/test/test_intrin128.sse4_1.cpp" "modules/core/test/test_intrin128.simd.hpp"
cLinks "build/modules/core/test/test_intrin128.sse4_1.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin128.sse4_2.cpp" "modules/core/test/test_intrin128.simd.hpp"
cLinks "build/modules/core/test/test_intrin128.sse4_2.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin128.ssse3.cpp" "modules/core/test/test_intrin128.simd.hpp"
cLinks "build/modules/core/test/test_intrin128.ssse3.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin256.avx2.cpp" "modules/core/test/test_intrin256.simd.hpp"
cLinks "build/modules/core/test/test_intrin256.avx2.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin256.avx512_skx.cpp" "modules/core/test/test_intrin256.simd.hpp"
cLinks "build/modules/core/test/test_intrin256.avx512_skx.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin256.simd_declarations.hpp" "modules/core/include/opencv2/core/private/
cLinks "build/modules/core/test/test_intrin512.avx512_skx.cpp" "modules/core/test/test_intrin512.simd.hpp"
cLinks "build/modules/core/test/test_intrin512.avx512_skx.cpp" "modules/core/test/test_precomp.hpp"
cLinks "build/modules/core/test/test_intrin512.simd_declarations.hpp" "modules/core/include/opencv2/core/private/
```

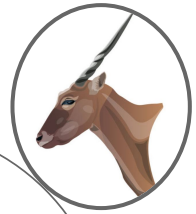
srcML Extraction Process

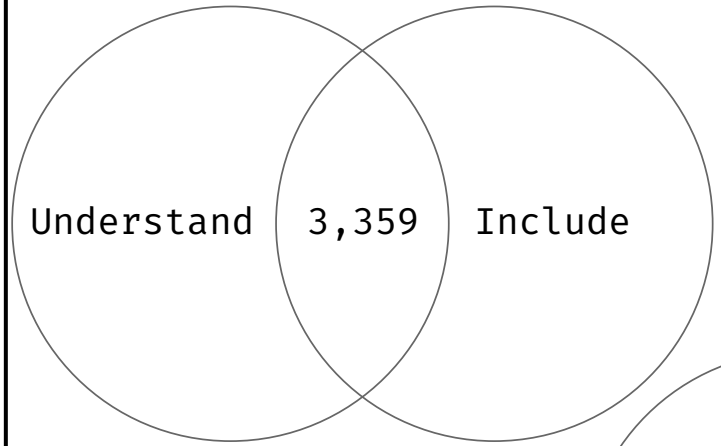


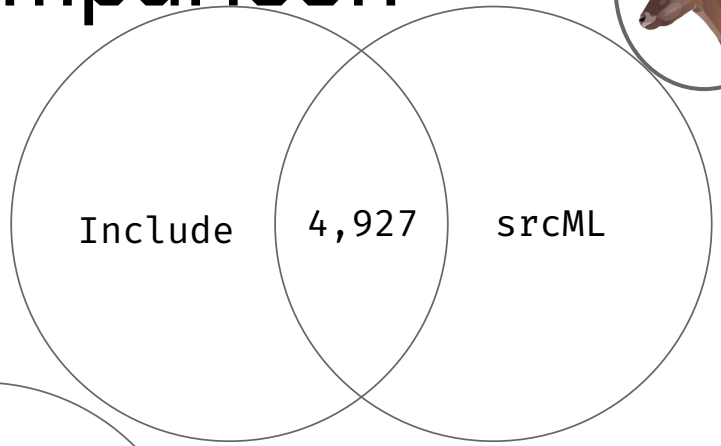
Quantitative Comparison

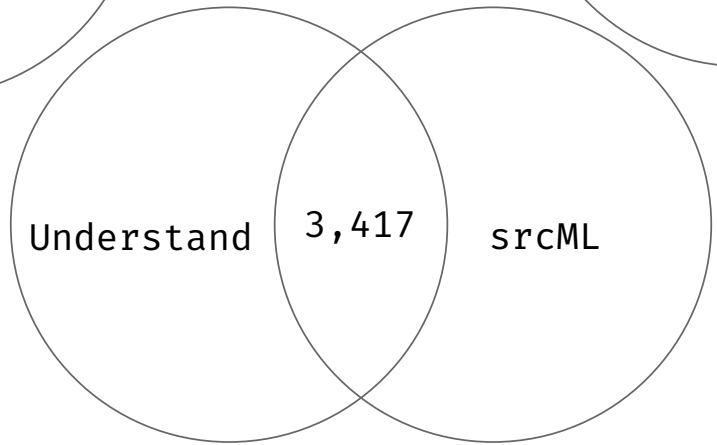
Dependency Technique	Total Dependencies
Understand	26,058
Include	31,041
srcML	9,280

Quantitative Comparison









Qualitative Analysis

Differences:

- Semantic-Level Extraction
- Structural-Level Extraction
- Syntactic-Level Extraction

Precision & Recall

$$\text{Precision} = \frac{\text{Edges found in tool(true)}}{\text{Edges found in tool(true) + Edges found in tool (false)}}$$

$$\text{Recall} = \frac{\text{Edges found in tool(true)}}{\text{Edges found in tool(true) + Edges NOT found in tool}}$$

Qualitative Analysis

Understand vs Include

Determine Sample Size

Confidence Level: (•) 95% ( ) 99%

Confidence Interval: 5

Population: 57099

Calculate Clear

Sample size needed: 382

Precision of Include = 27.78%

Recall of Include = 35%

Qualitative Analysis

Understand vs srcML

Determine Sample Size

Confidence Level: (•) 95% ( ) 99%

Confidence Interval: 5

Population: 35338

Calculate Clear

Sample size needed: 380

Precision of srcML = 66.67%

Recall of srcML = 11.11%

Qualitative Analysis

Include vs srcML

Determine Sample Size

Confidence Level: (•) 95% ( ) 99%

Confidence Interval: 5

Population: 40321

Calculate Clear

Sample size needed: 381

Precision of srcML = 42.17%

Recall of srcML = 10.51%

Limitations of Our Findings

Lessons Learned

1. Include extraction is fast
2. Understand excels in full-architecture extraction
3. A hybrid approach may be ideal

Conclusion

Questions?