Structure from Motion in Mojo🔥

Taylor Pool, Easton Potokar

What is Mojo🔥?

"Feels like Python, runs like C"

Mojo is a statically-typed (read: fast) language that aims to be a *superset* of python (read: easy)

It attempts solve the two-language problem.

Project Goal:
1. Evaluate Mojo's claims on CV problems
2. Reimplement SfM in a straightforward manner for others.

```
@value
@register_passable("trivial")
struct Landmark:
    var val: mc.Vector3d

    @always_inline
    @staticmethod
    fn dim() -> Int:
        return 3

    @always_inline
    @staticmethod
    fn identity() -> Self:
        return Self {val: mc.Vector3d(0, 0, 0, 0)}

    fn __invert__(self) -> Self:
        return Self(-self.val)

    fn inv(self) -> Self:
        return ~self
```

Mojo Pros

SPEED

Static Executable Generation

```
(moca) tpool@taylor-dell-laptop:~/mojo_ws/tmp$ mojo build main.mojo
(moca) tpool@taylor-dell-laptop:~/mojo_ws/tmp$ ls
main  main.mojo
(moca) tpool@taylor-dell-laptop:~/mojo_ws/tmp$ ./main
Hello, World!
```

REPL

```
(moca) tpool@taylor-dell-laptop:~/mojo_ws/moca$ mojo
Welcome to Mojo! 🔥

Expressions are delimited by a blank line.
Type `:quit` to exit the REPL and `:mojo help` for further assistance.

  1> print("Hello, World!")
  2.
Hello, World!
  2> 1.0+2.0
  3.
  3> %%python
  4. import numpy as np
  5. x = np.arange(4)
  6. print(x)
  7.
[0 1 2 3]
```

Python Interop/Similar Syntax

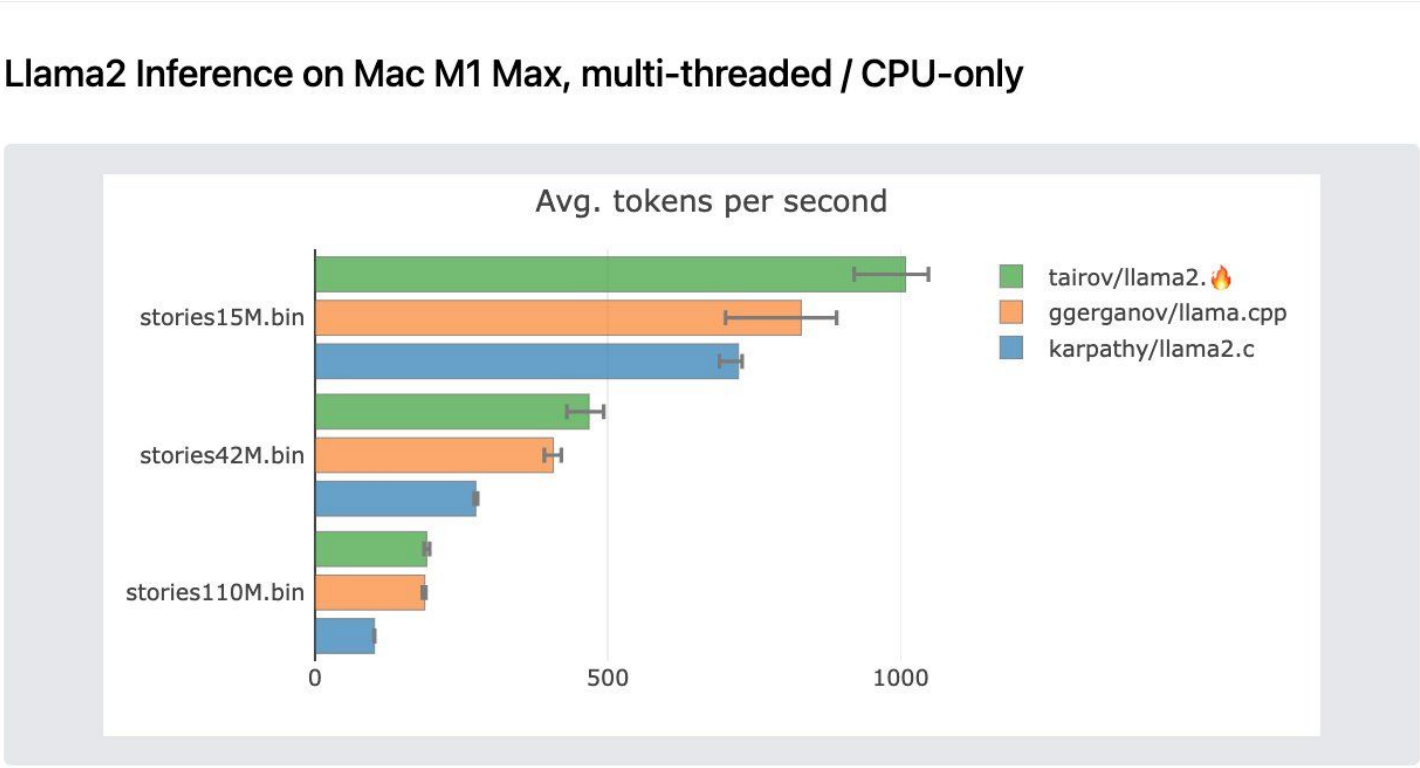
```mojo
from python import Python

def main():
    np = Python.import_module("numpy")
    x = np.arange(4)
    y = np.ones_like(x)
    z = np.matmul(x, y)
    print(z)
```

Missing in Mojo

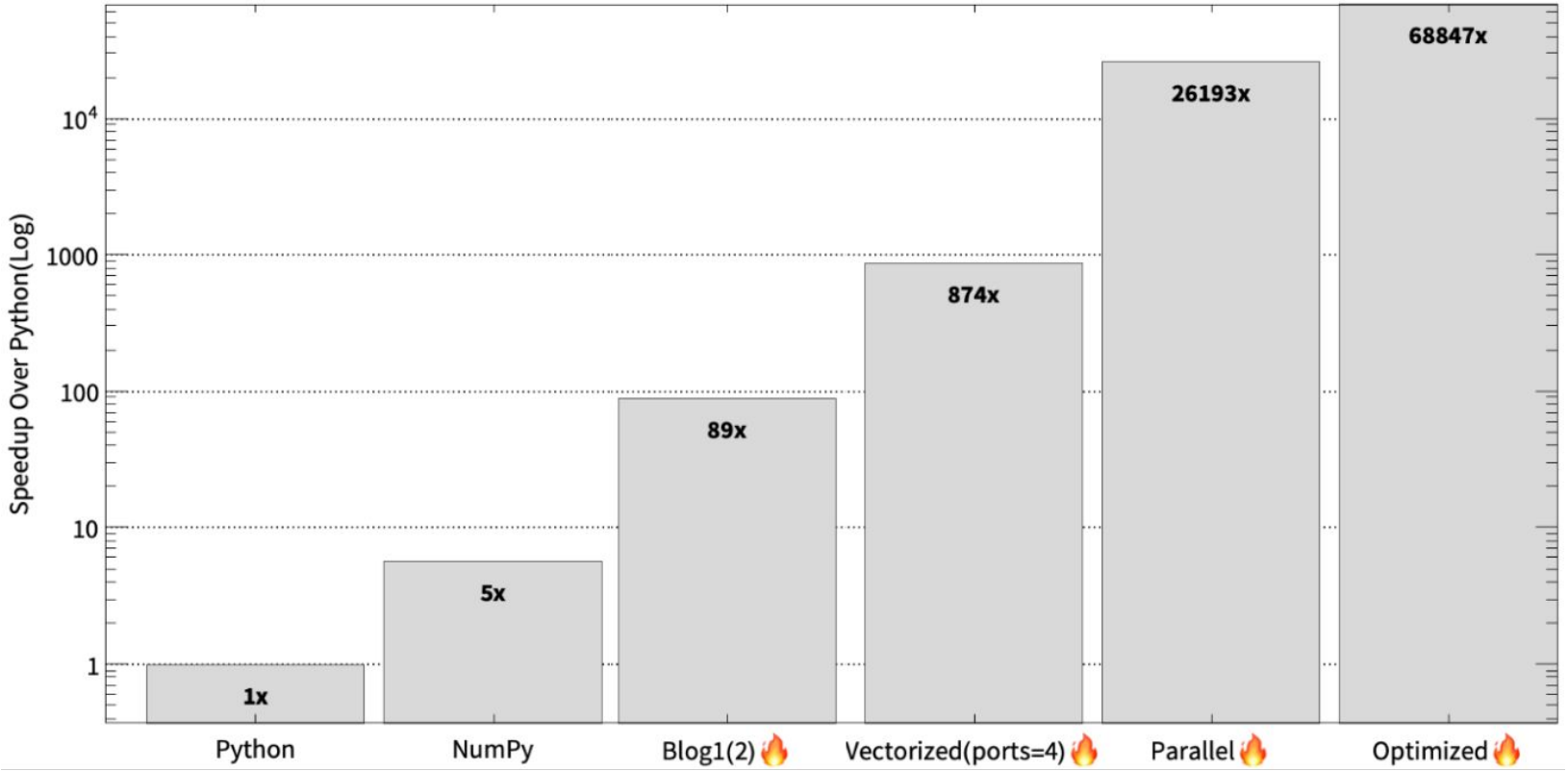
- Static Inheritance
- Borrow Checker
- Set/Dictionary
- Linear Algebra
- No package manager

```
@value      Easton Potokar, 18 hours ago • Getting closer....
struct Set[type: AnyType, hash: fn (type) -> Int]:
    var n: Int
    var elements: DynamicVector[type]
    var exists: Tensor[DType.bool]

    fn __init__(inout self, n: Int):
        self.n = n
        self.elements = DynamicVector[type](n)
        self.exists = Tensor[DType.bool](n)

    fn add(inout self, x: type):
        let e = hash(x)
        if self.exists[e]:
            return
        else:
            self.elements.push_back(x)
            self.exists[e] = True

    fn contains(self, x: type) -> Bool:
        let e = hash(x)
        return self.exists[e]
```

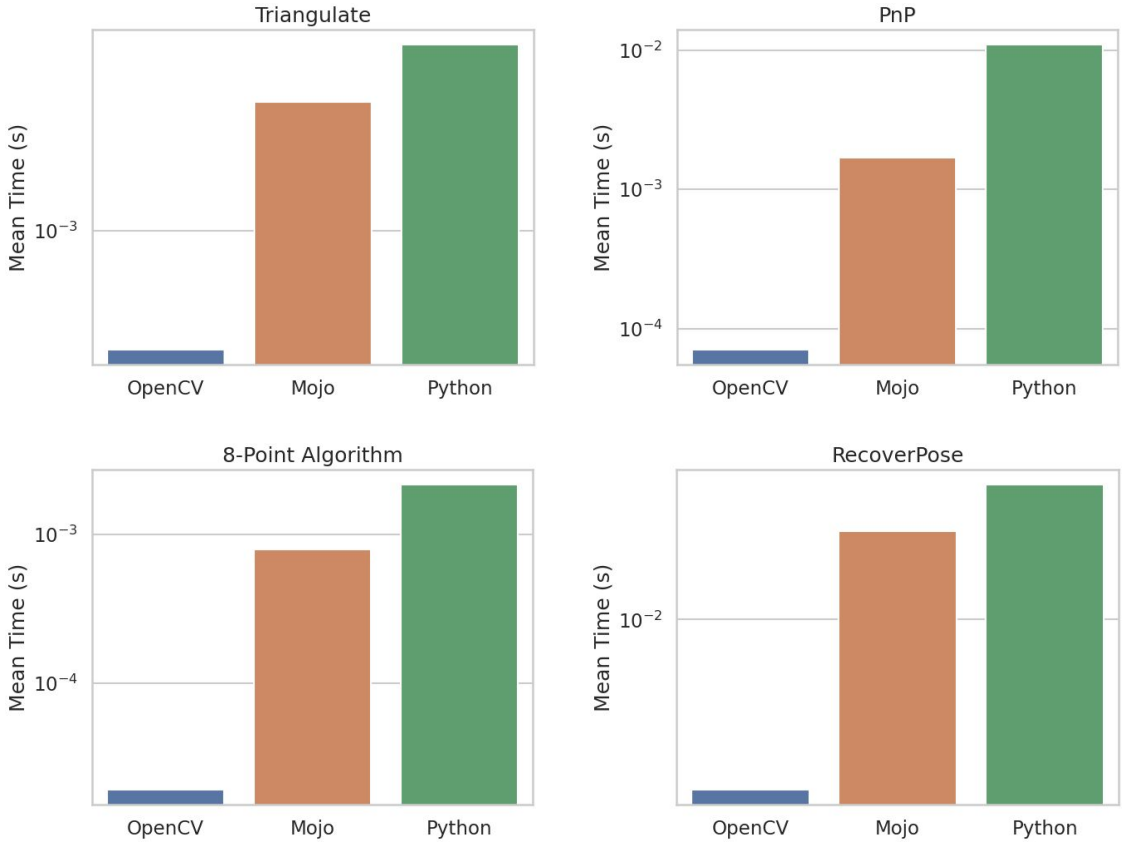
Benchmarks - Llama2 (Others)



Benchmarks - Mandelbrot (Others)



Benchmarks - Computer Vision (Ours)



Benchmarks - Interpretations

Why weren’t we seeing such large improvements?

Naive linear algebra implementations

```
224  fn mat_mat[type: DType](lhs: Tensor[type], rhs: Tensor[type]) -> Tensor[type]:
225      let m = lhs.shape()[0]
226      let n = rhs.shape()[1]
227      let p = rhs.shape()[0]
228      var result = Tensor[type](m, n)
229      for i in range(m):
230          for j in range(n):
231              let i_j = Index(i, j)
232              for k in range(p):
233                  result[i_j] += lhs[i, k] * rhs[k, j]
234
235      return result
```

Our Structure from Motion Implementation

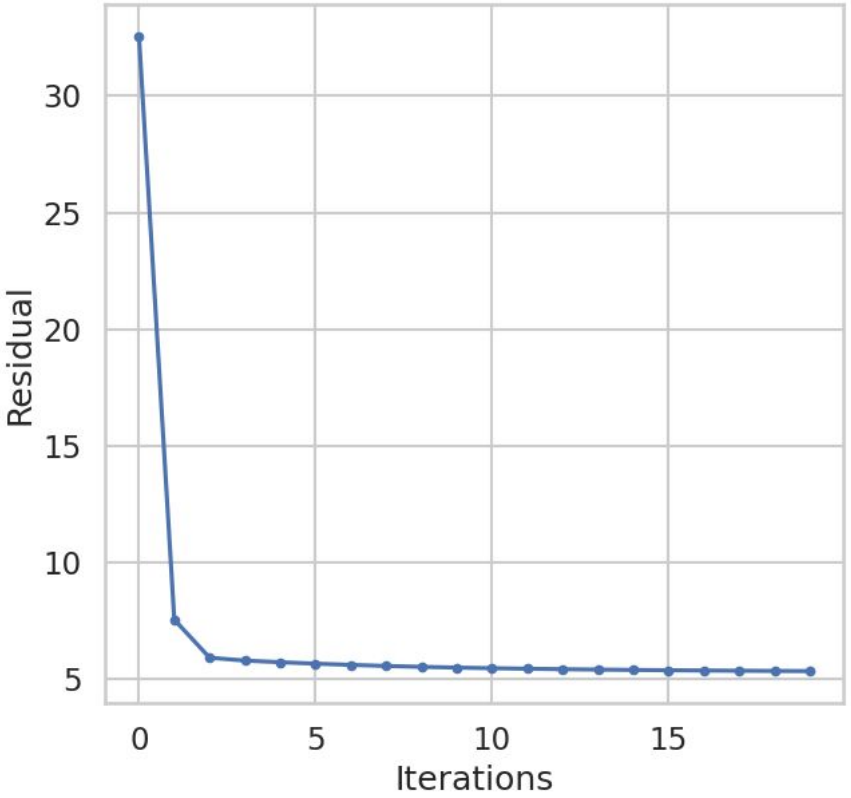
Used COLMAP for scene graph generation

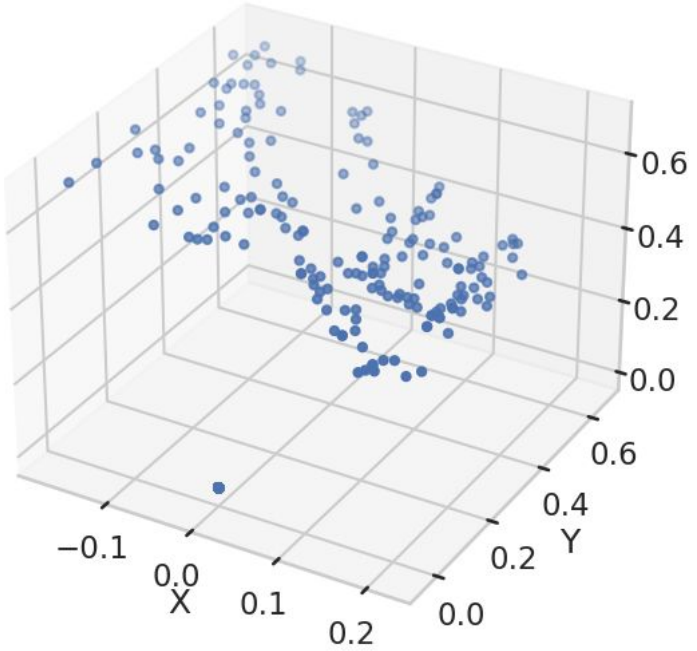
Our backend implementation includes:

- SE3 / SO3 implementation
- Basic linear algebra library
- CV estimation functions
  - Fundamental & essential matrix, PnP, triangulation
- Levenberg-Marquardt
- All necessary bookkeeping for SfM (nontrivial!)
- Full test coverage of all systems

taylorpool/moca

Preliminary Results





Key Takeaways

Mojo isn’t ready for primetime… yet. However,

- It really is easier & faster to code in than C++
- You *can* import any python code, but interoperability with python objects is difficult at the moment
- Eventually may be a better option than rather a split C++/Python codebase

Also

- SfM bookkeeping is *hard.* Like, really nontrivial.