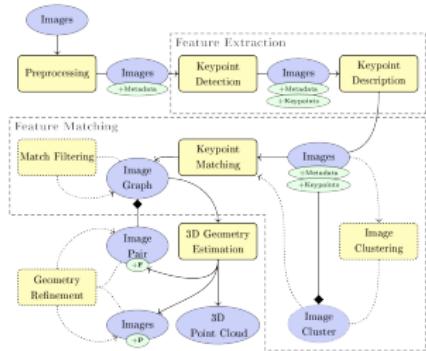


Master's thesis

A Modular Framework for Image-based 3D Reconstruction



Carsten Brandt

May 18, 2017



3D Reconstruction

3D Reconstruction

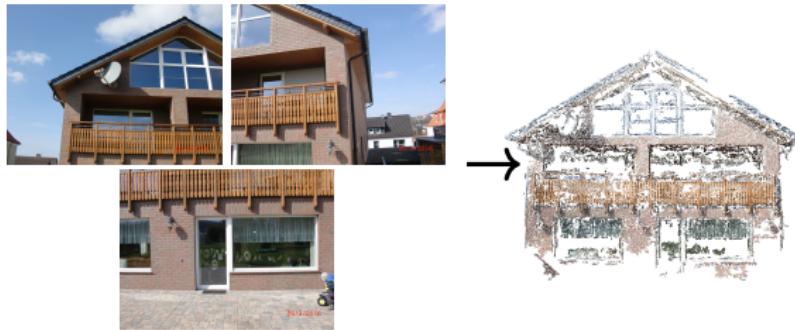
Structure from Motion (SfM)

3D Reconstruction



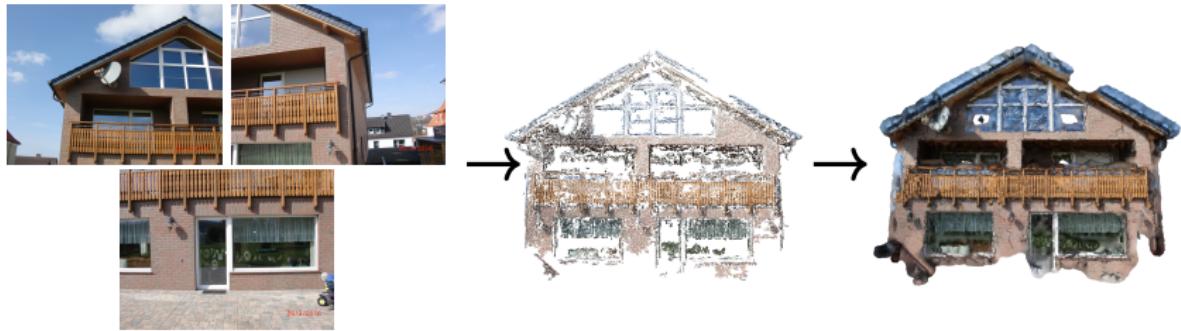
Structure from Motion (SfM)

3D Reconstruction



Structure from Motion (SfM)

3D Reconstruction



Structure from Motion (SfM)

Table of Contents

1 Problem Definition

2 The Process of 3D Reconstruction

3 Framework Design and Implementation

4 Using the Framework for SfM

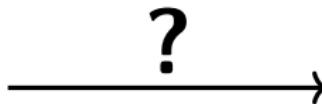
Table of Contents

- 1 Problem Definition
- 2 The Process of 3D Reconstruction
- 3 Framework Design and Implementation
- 4 Using the Framework for SfM

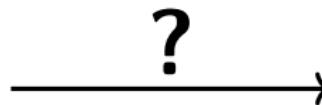
Problem Definition



Problem Definition



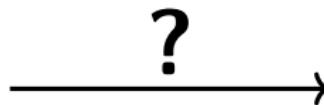
Problem Definition



Keypoint Detection



Problem Definition

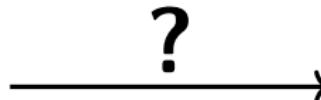


Keypoint Detection

Image Matching



Problem Definition



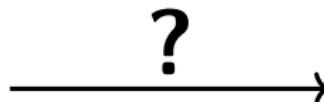
Keypoint Detection

Image Matching

Geometry Estimation



Problem Definition



Keypoint Detection

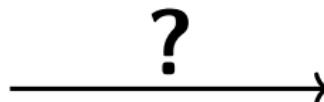
Image Matching

Geometry Estimation

Triangulation



Problem Definition



Keypoint Detection

Image Matching

Geometry Estimation

Triangulation

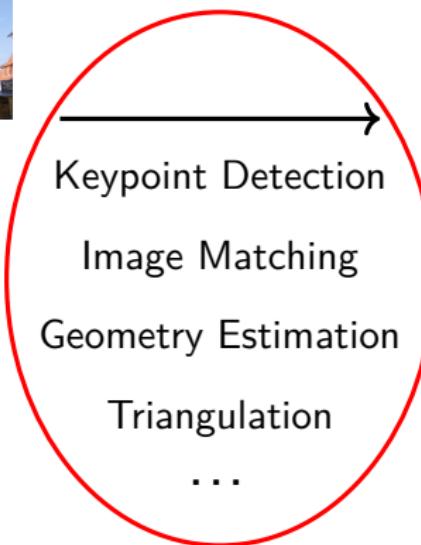
...



Problem Definition



Processing Chain



Existing implementations

Existing implementations:

Existing implementations

Existing implementations:

- single program for everything

Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case

Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve

Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve



Existing implementations

Existing implementations: Framework goals:

- single program for everything
- not flexible for different use case
- hard to adjust/improve



Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve

Framework goals:

- flexible processing chain creation



Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve



Framework goals:

- flexible processing chain creation
- applicable to large reconstruction problems

Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve



Framework goals:

- flexible processing chain creation
- applicable to large reconstruction problems
- visualisation options

Existing implementations

Existing implementations:

- single program for everything
- not flexible for different use case
- hard to adjust/improve



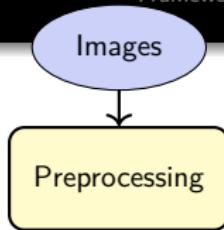
Framework goals:

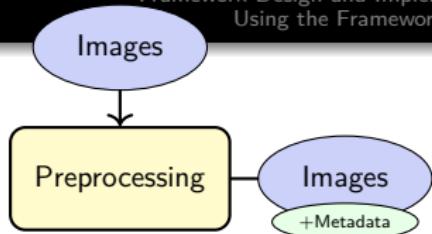
- flexible processing chain creation
- applicable to large reconstruction problems
- visualisation options

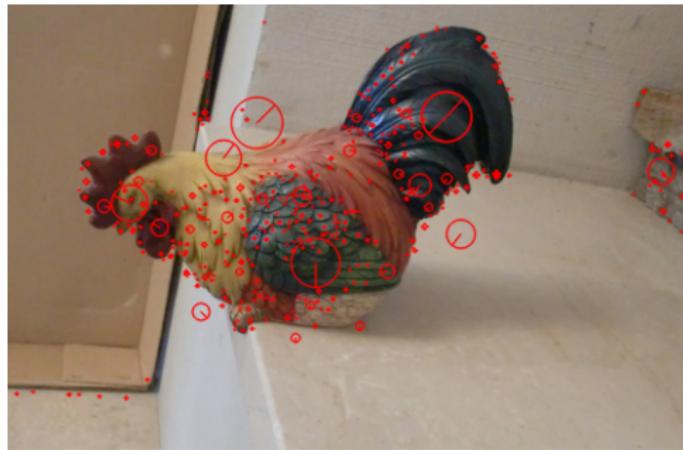
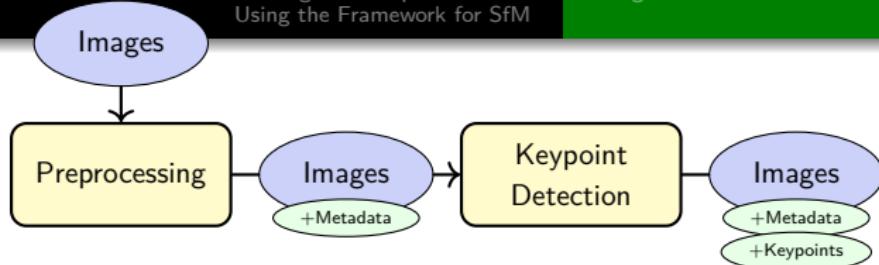


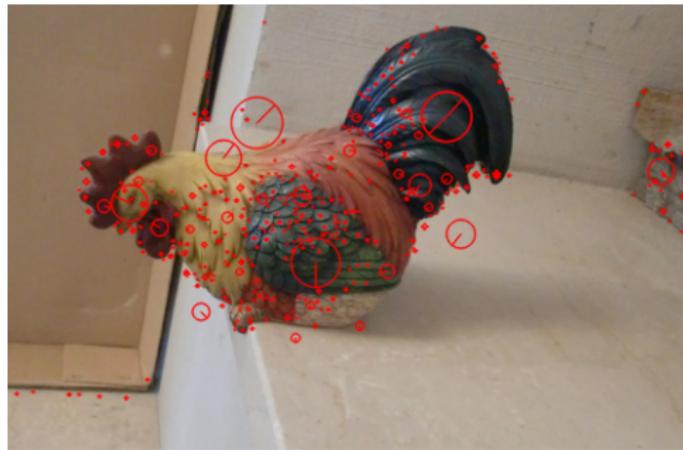
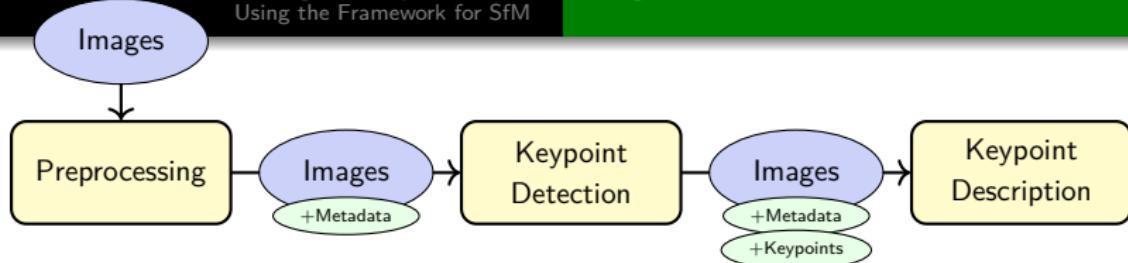
Table of Contents

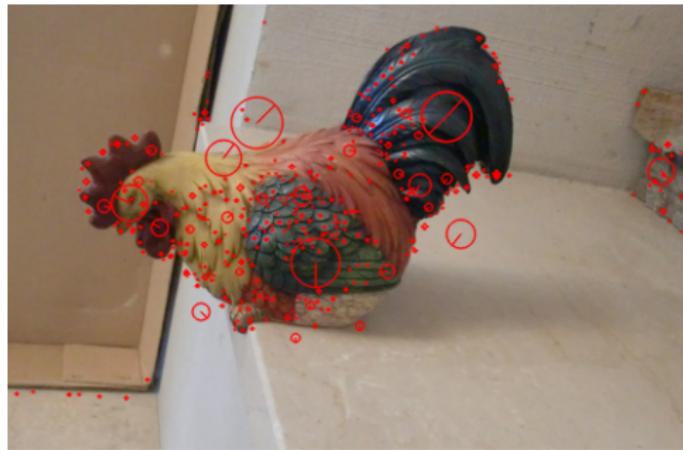
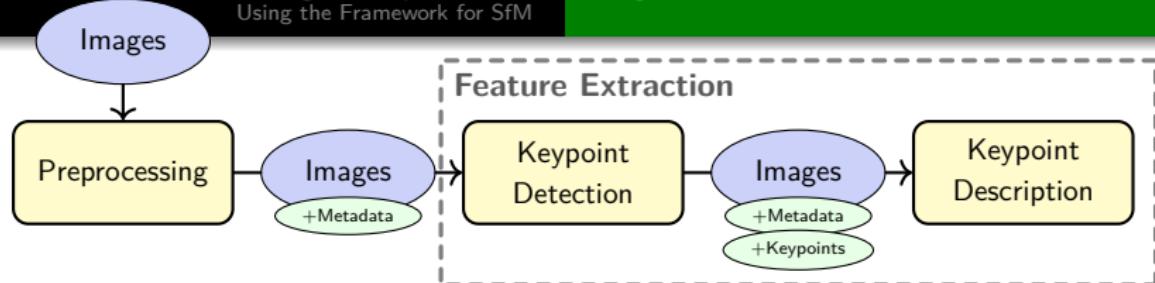
- 1 Problem Definition
- 2 The Process of 3D Reconstruction
- 3 Framework Design and Implementation
- 4 Using the Framework for SfM

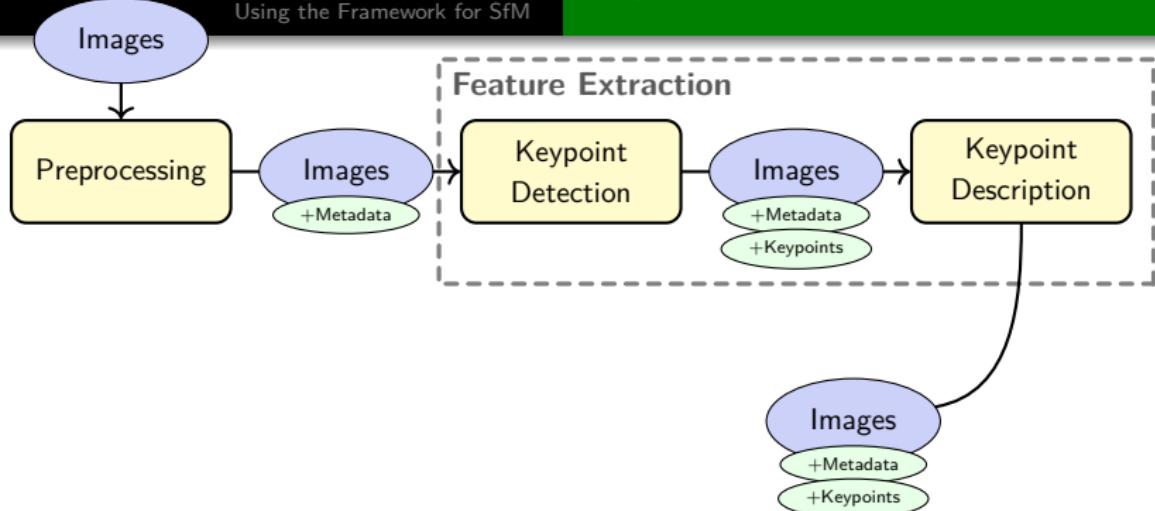


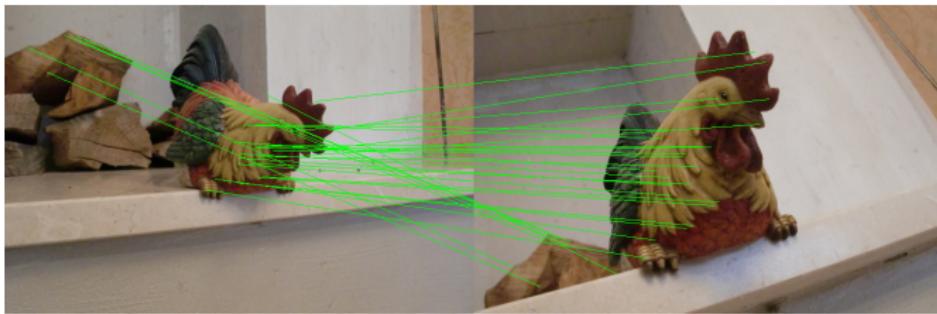
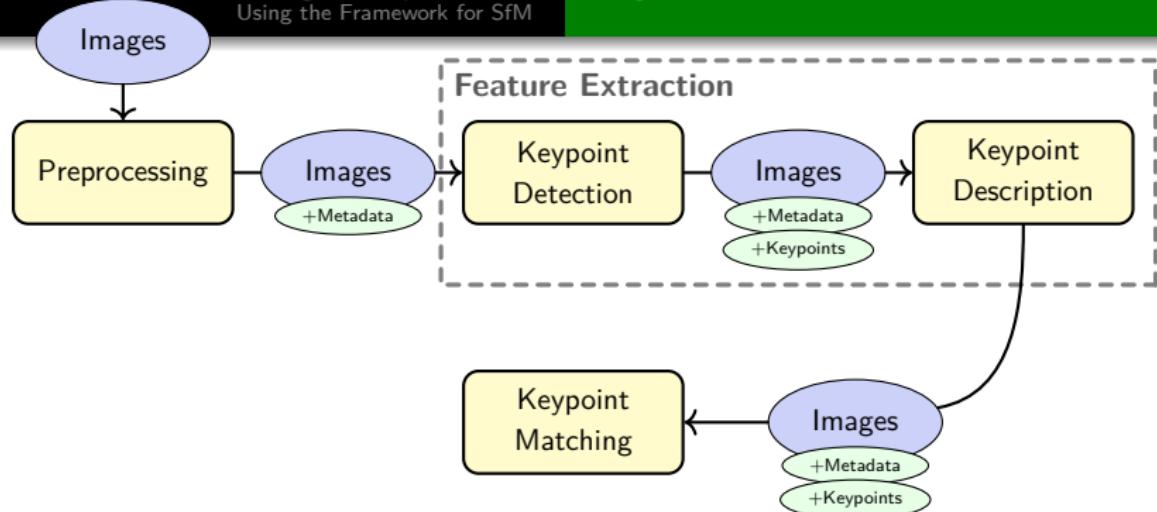


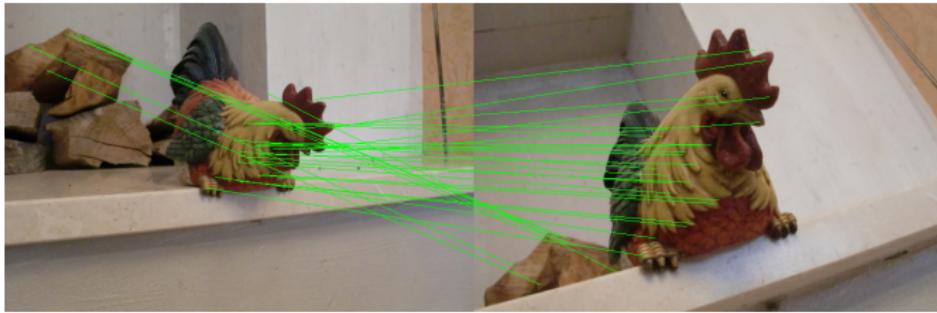
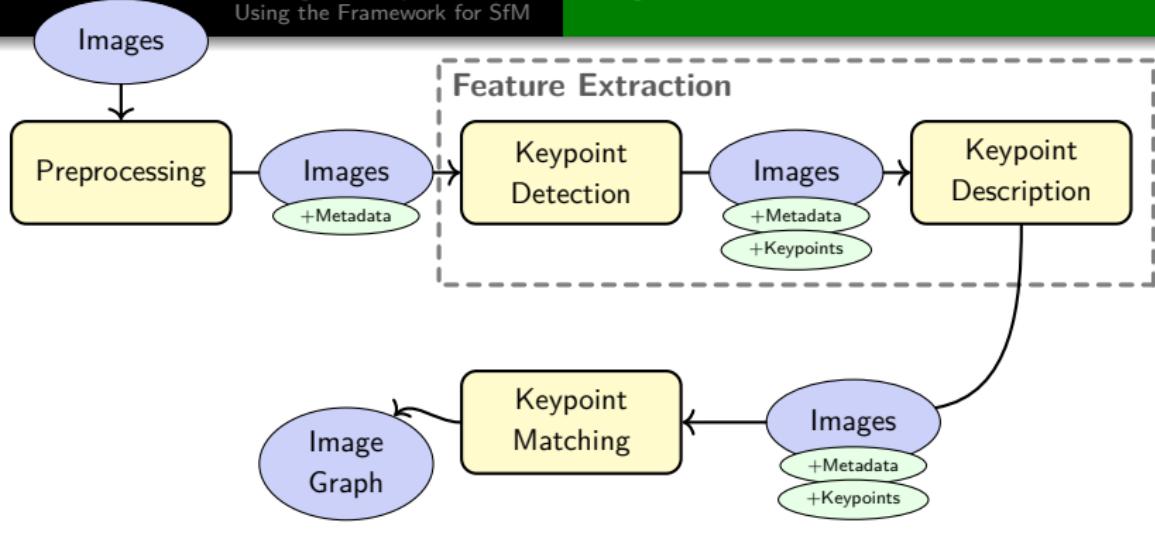


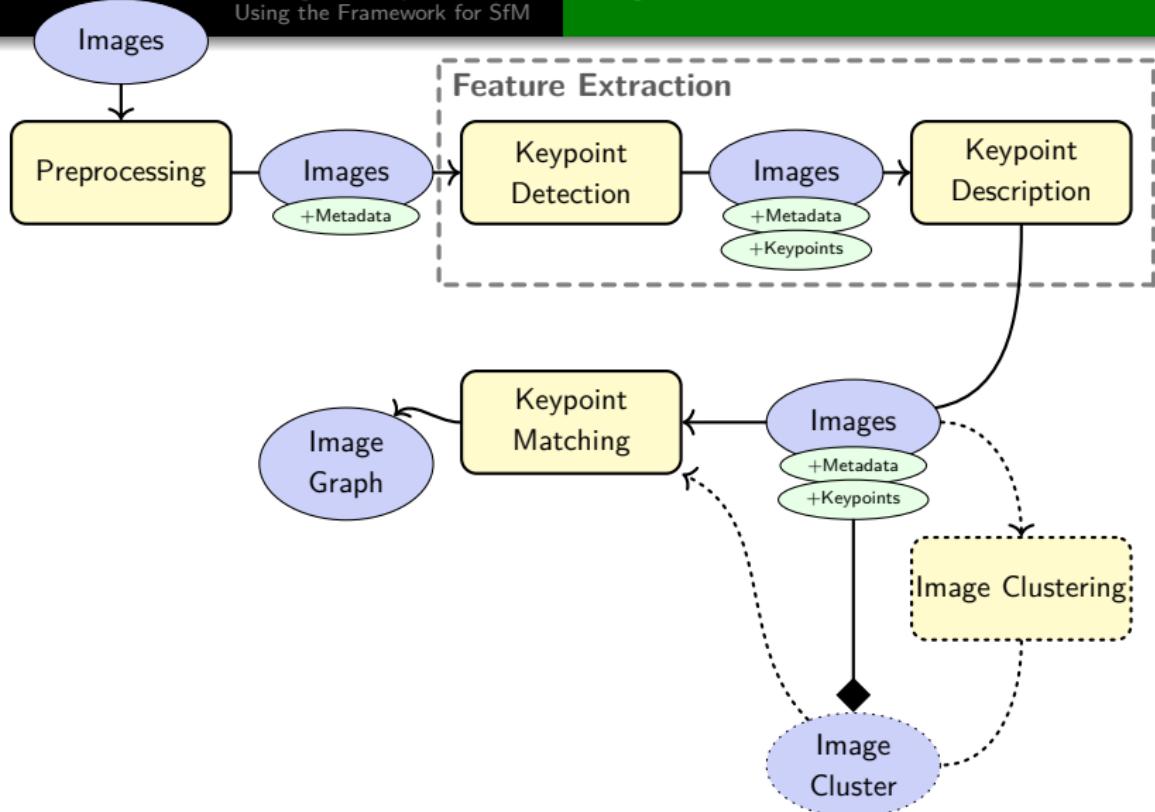


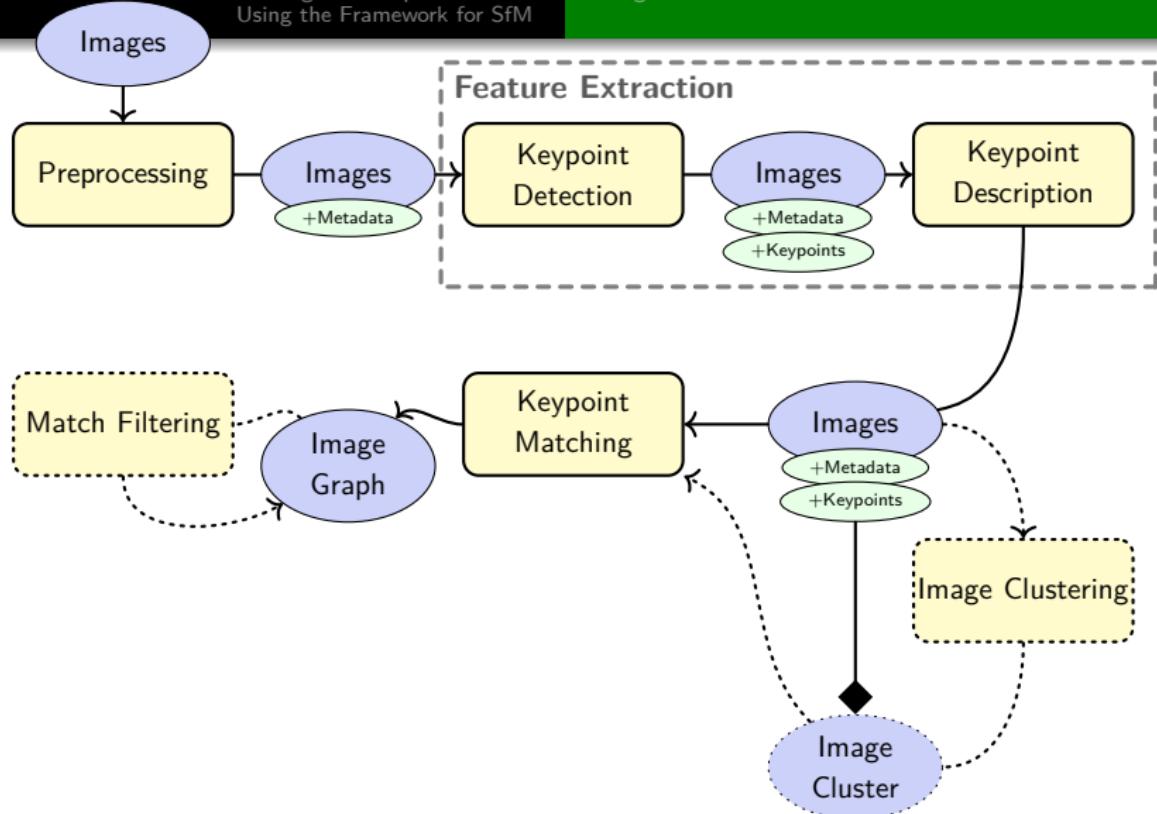


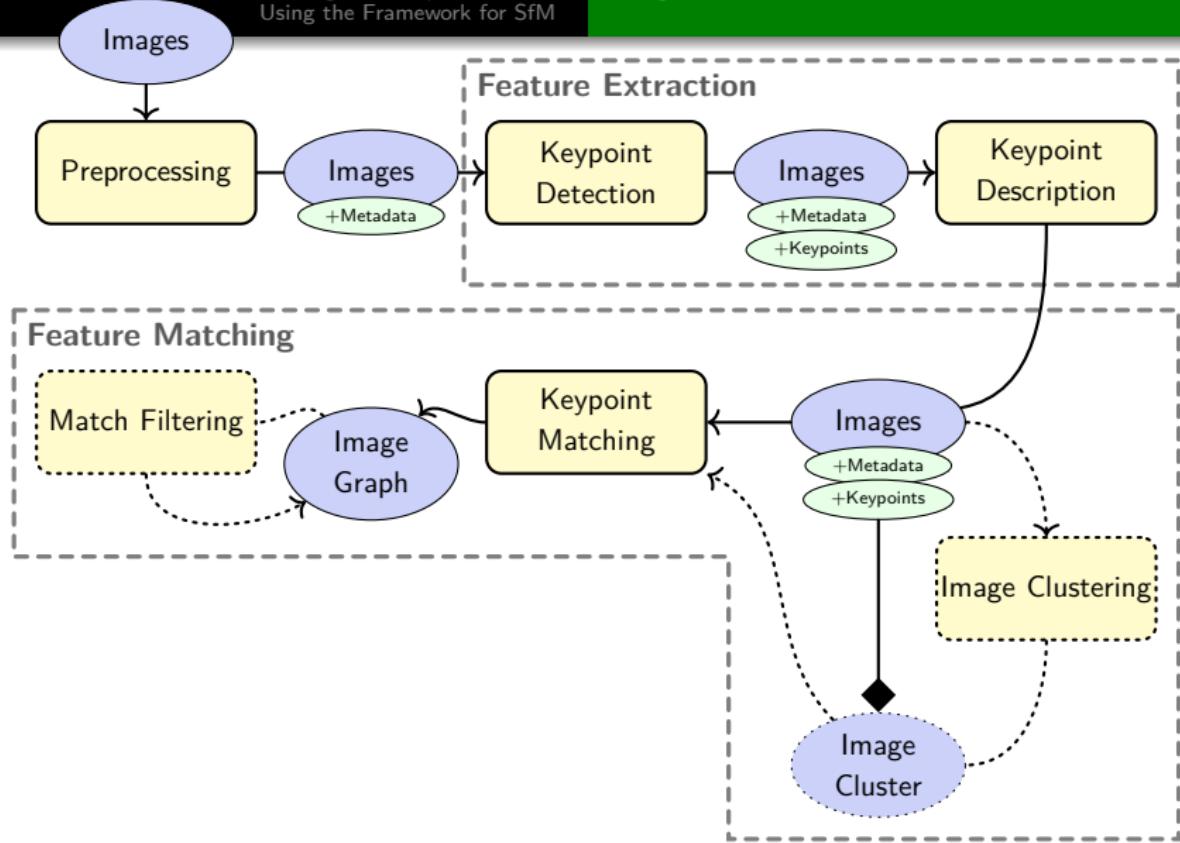


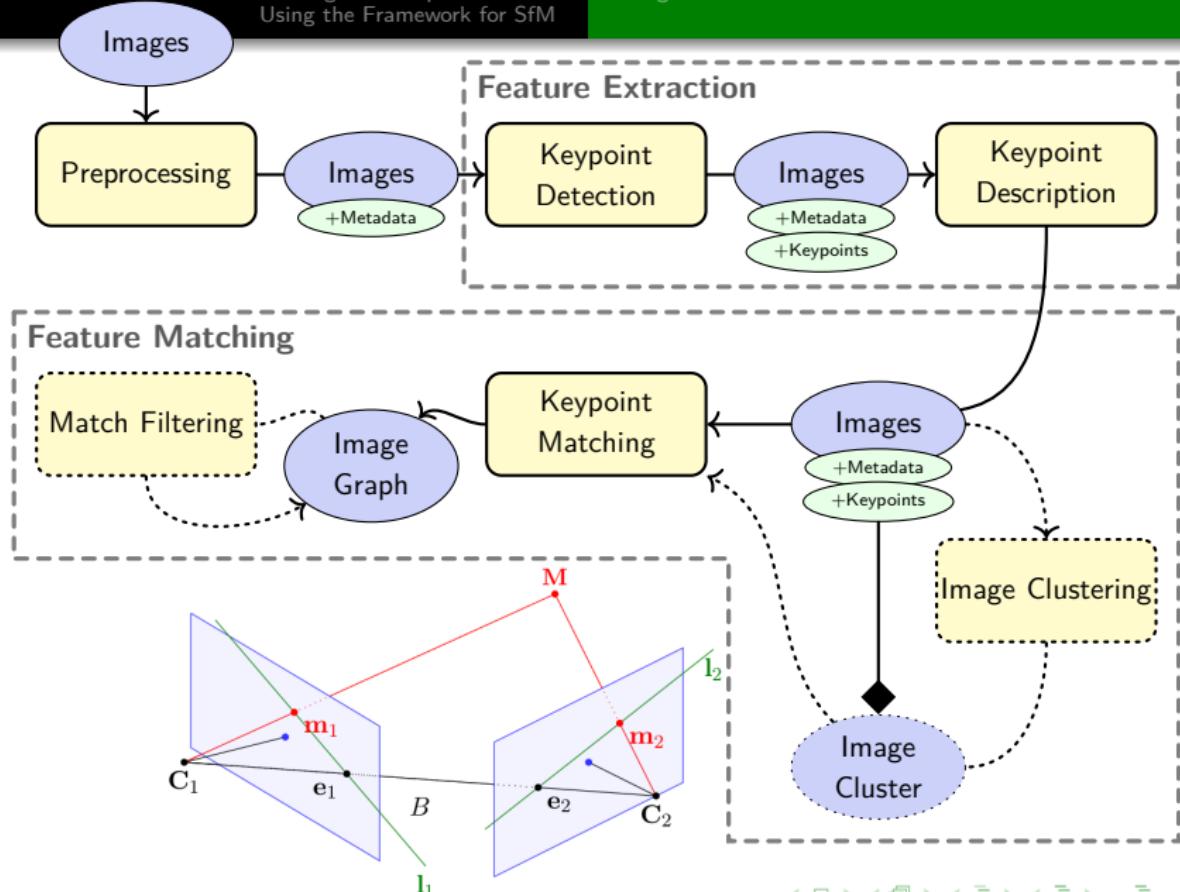


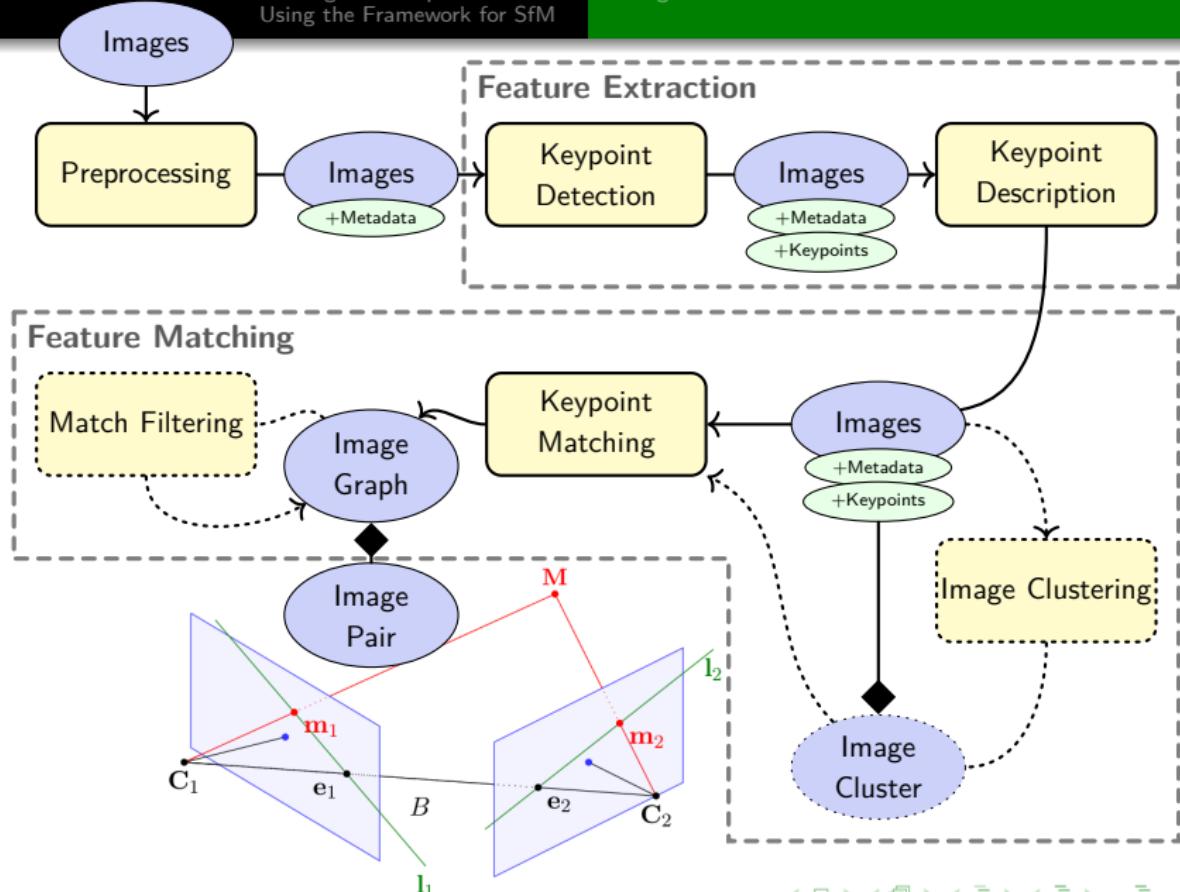


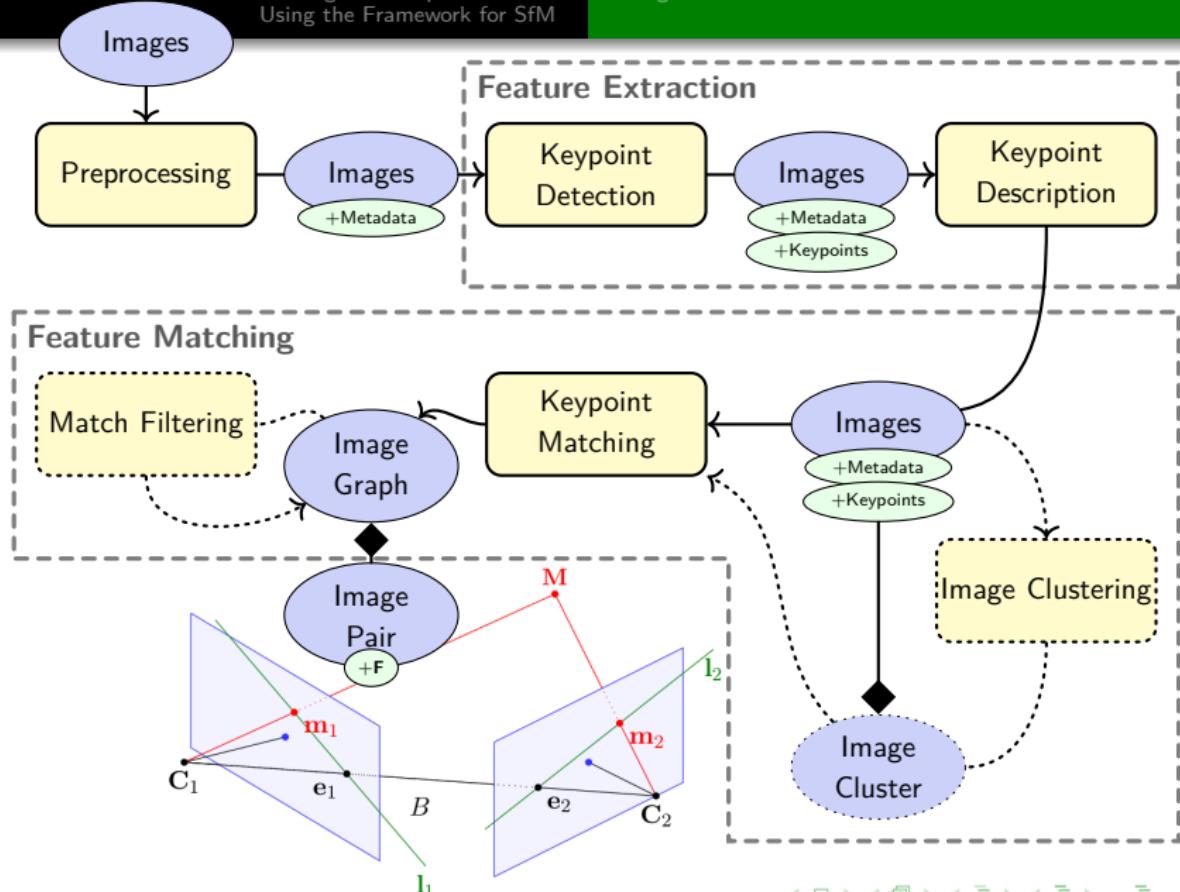


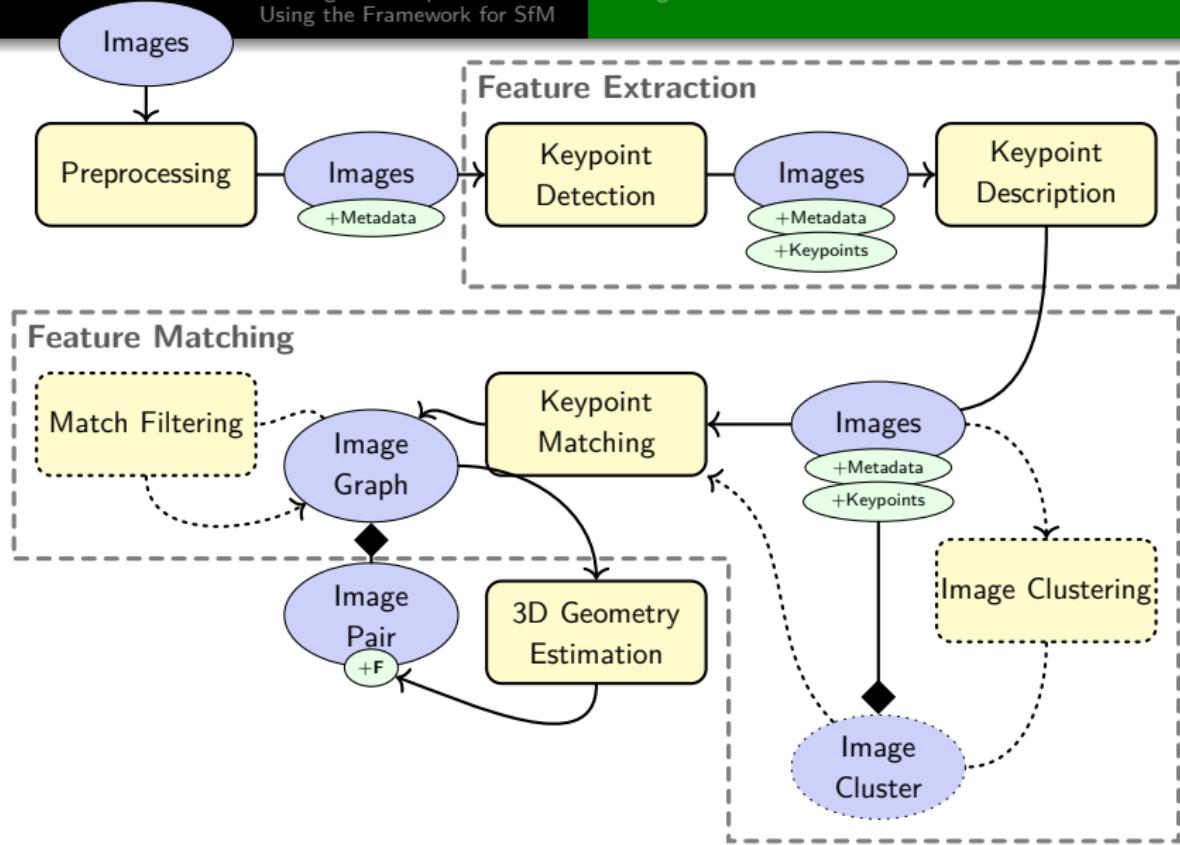


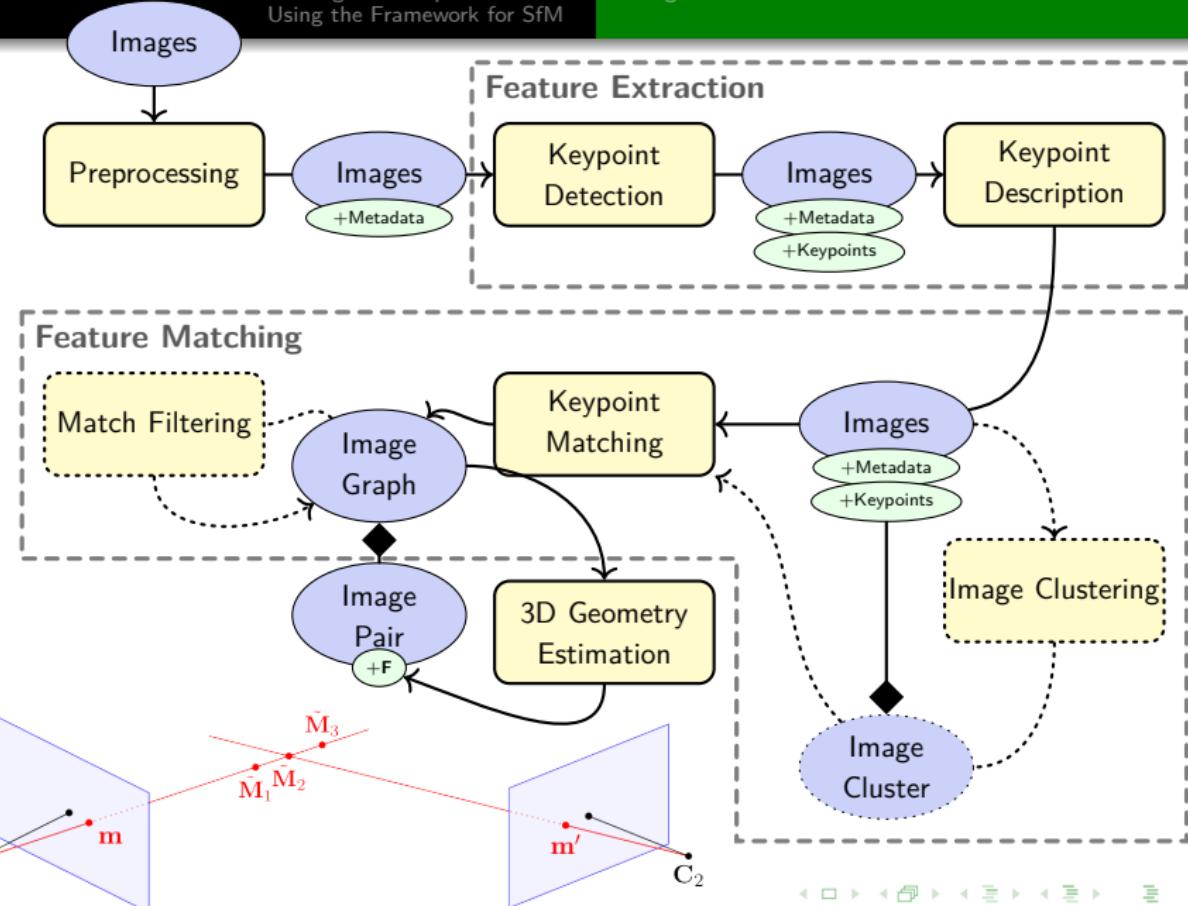


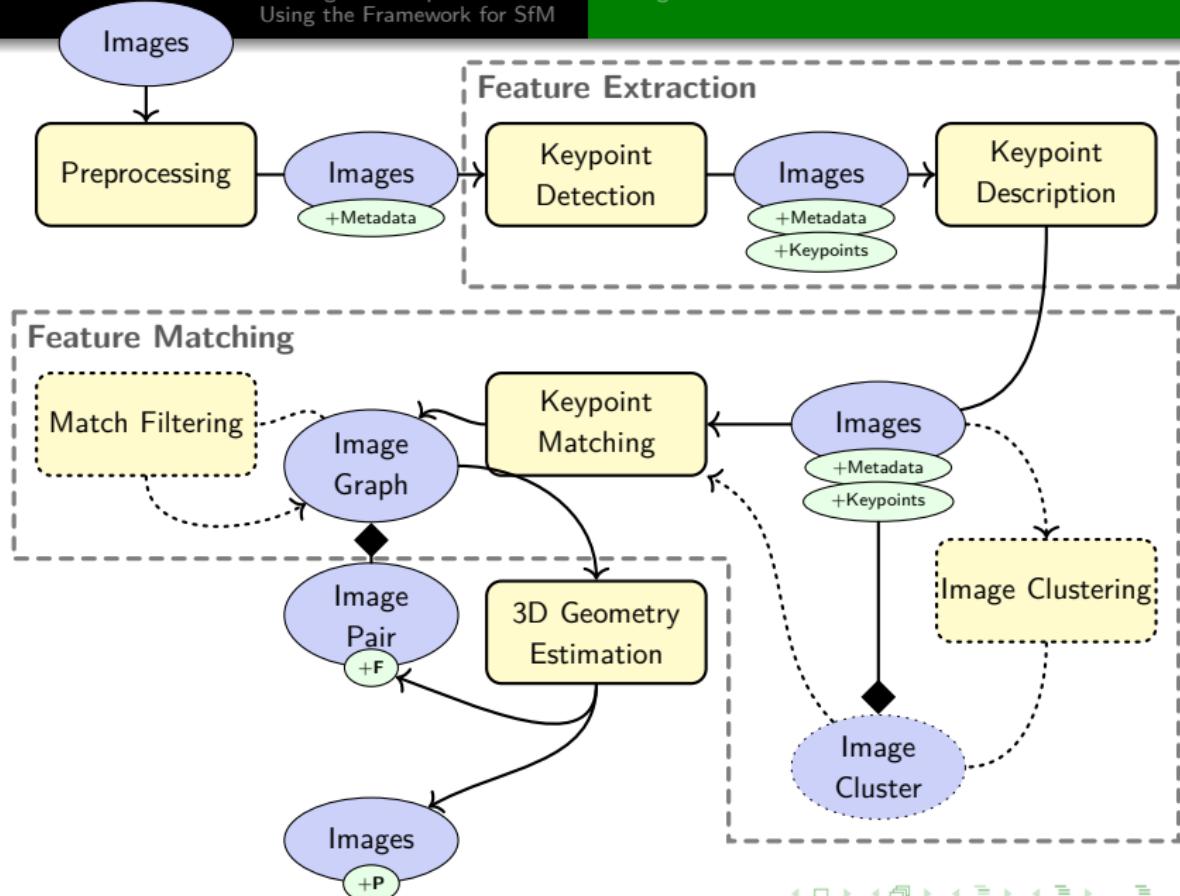


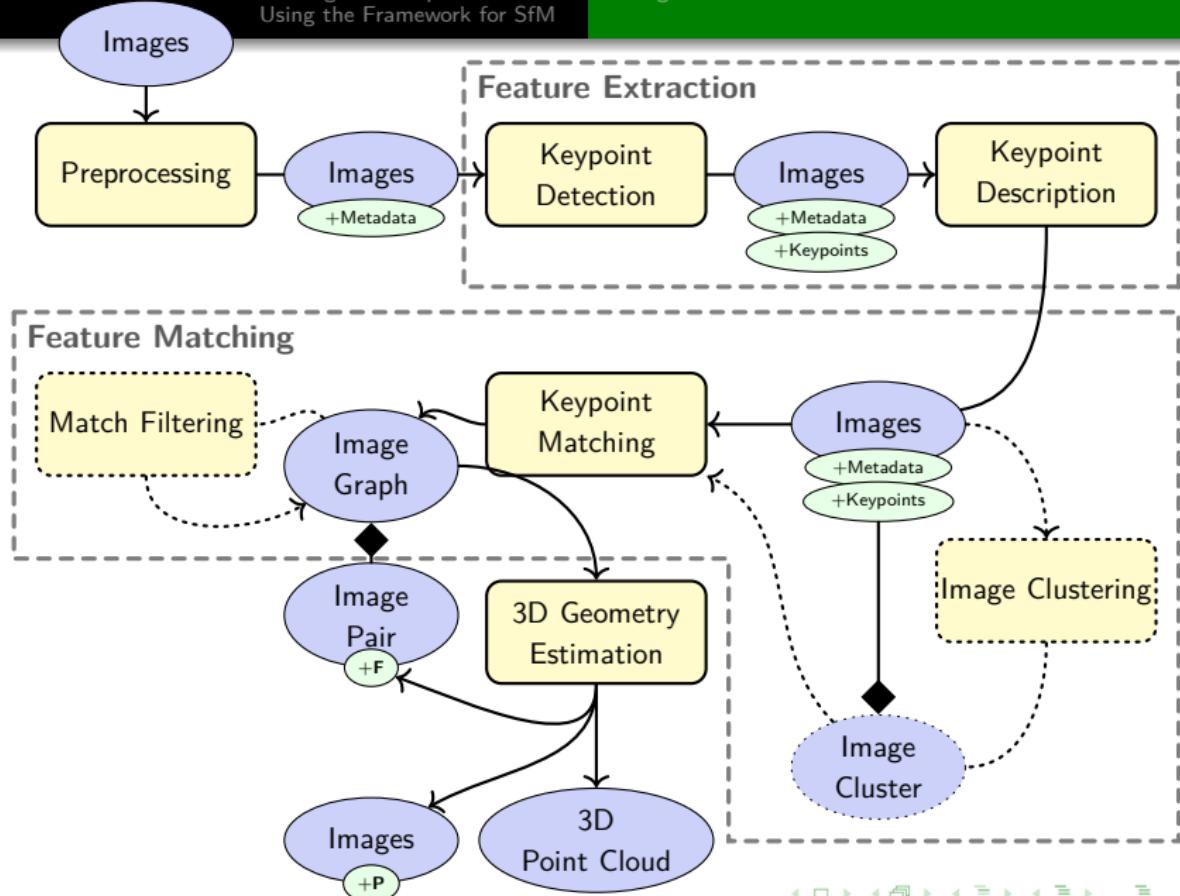


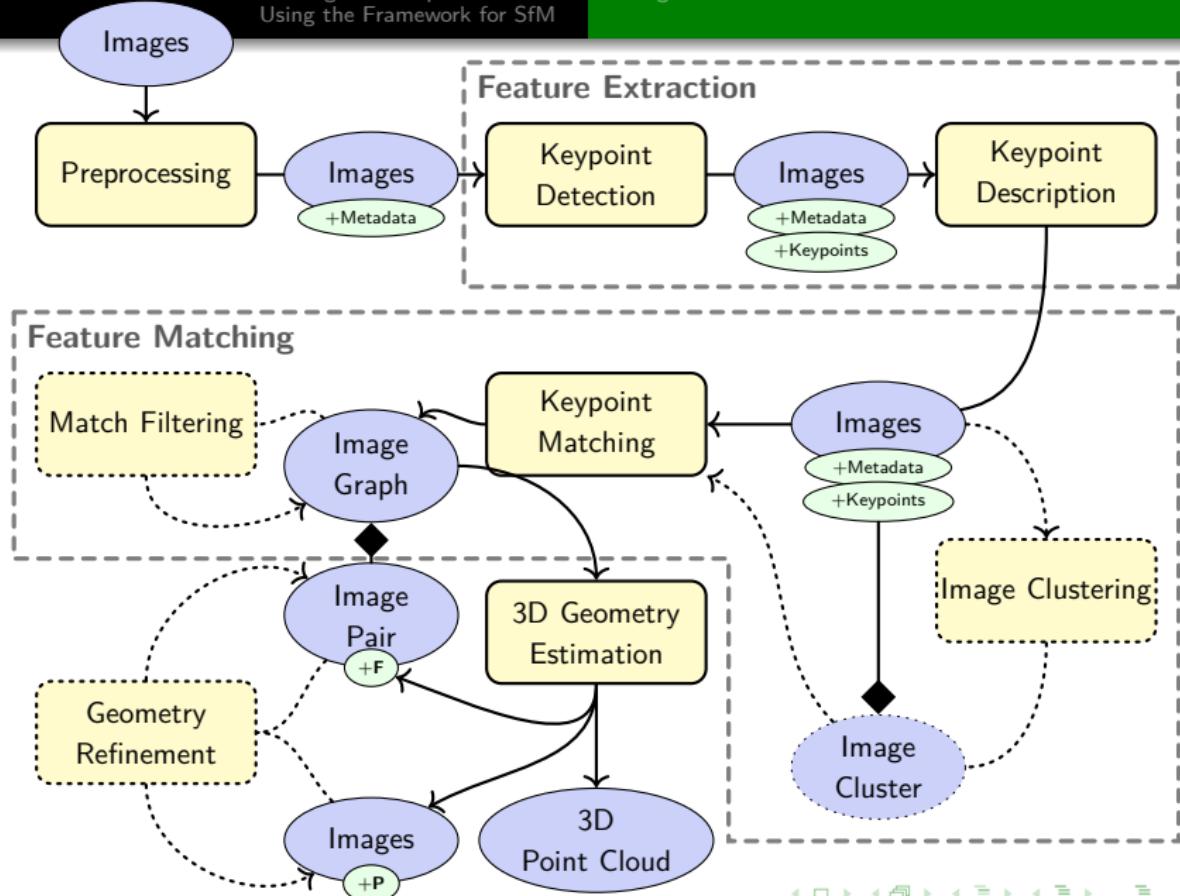




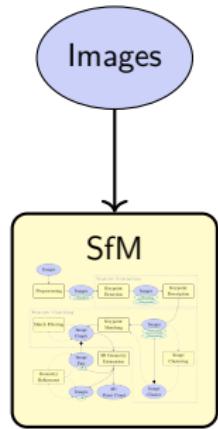




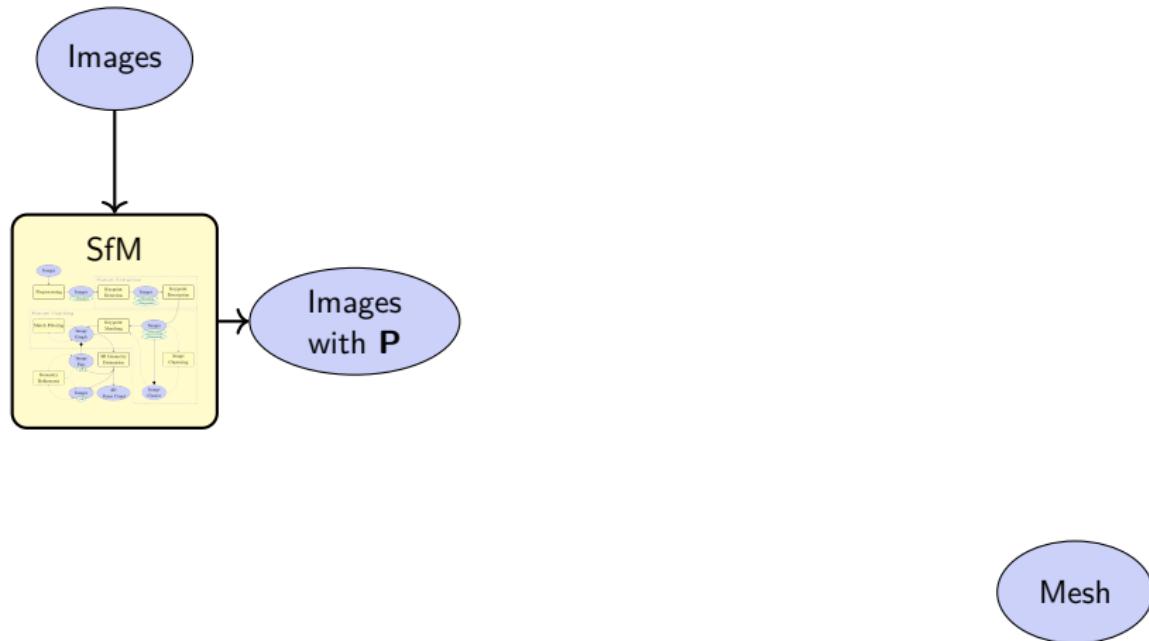




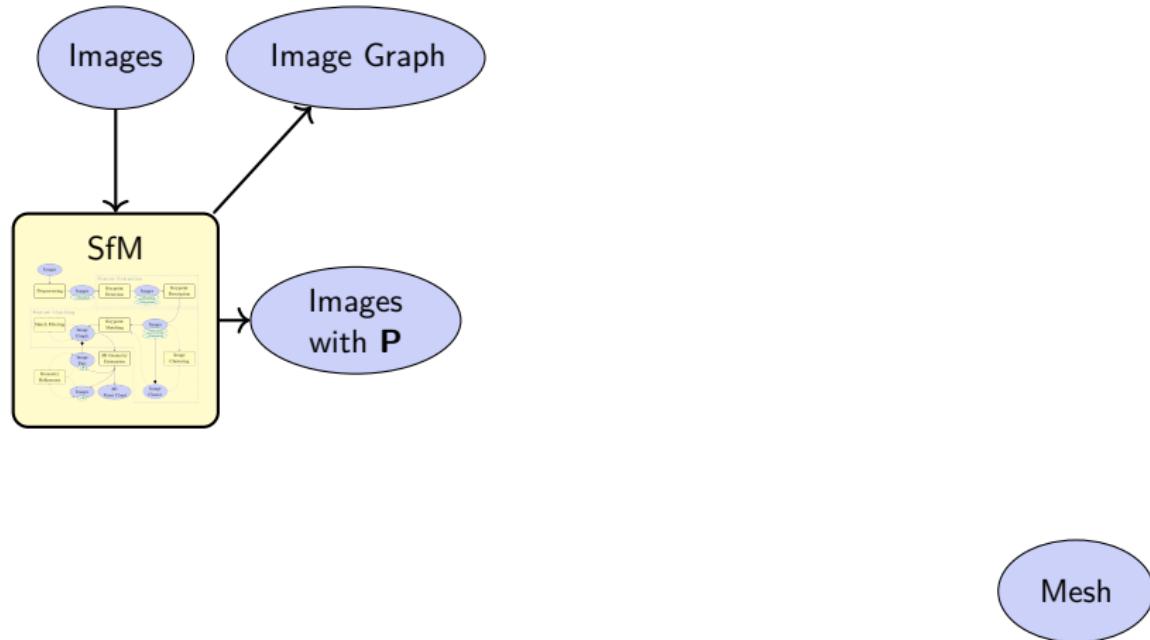
Highlevel overview of image-based 3D reconstruction



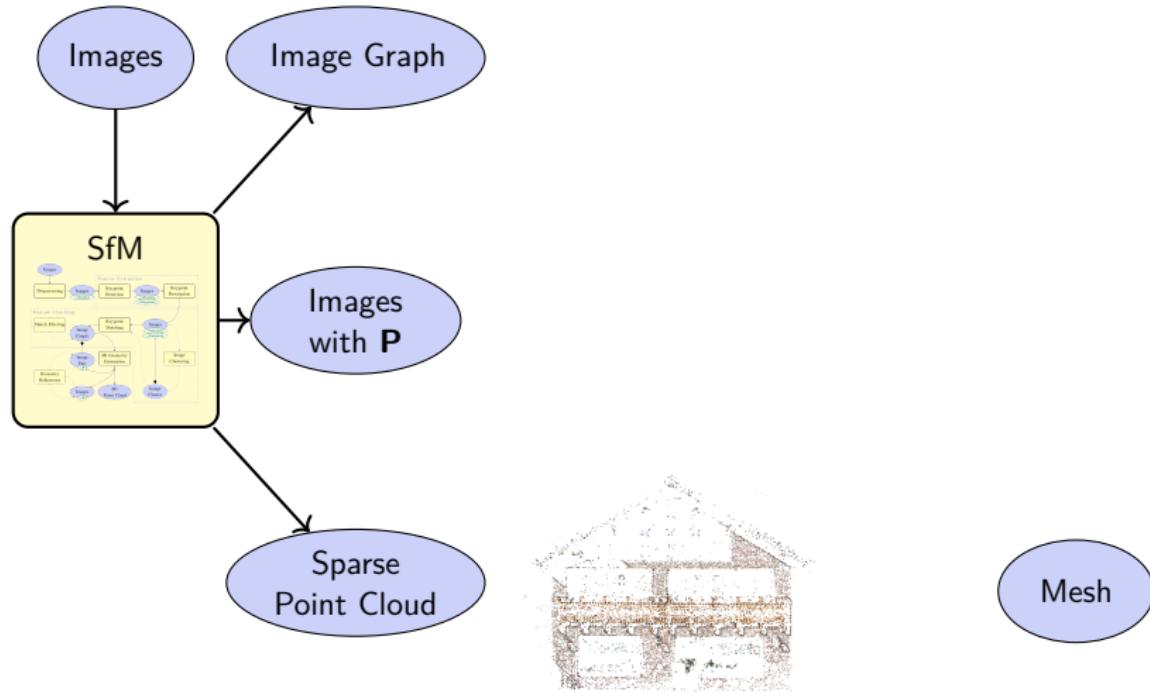
Highlevel overview of image-based 3D reconstruction



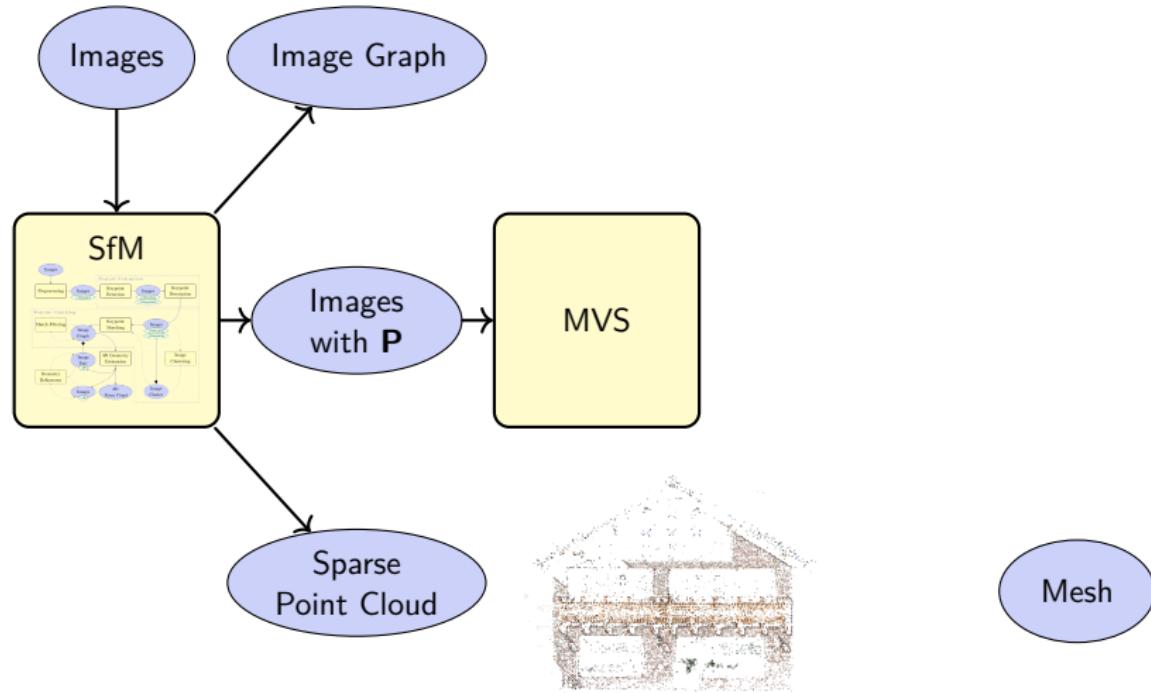
Highlevel overview of image-based 3D reconstruction



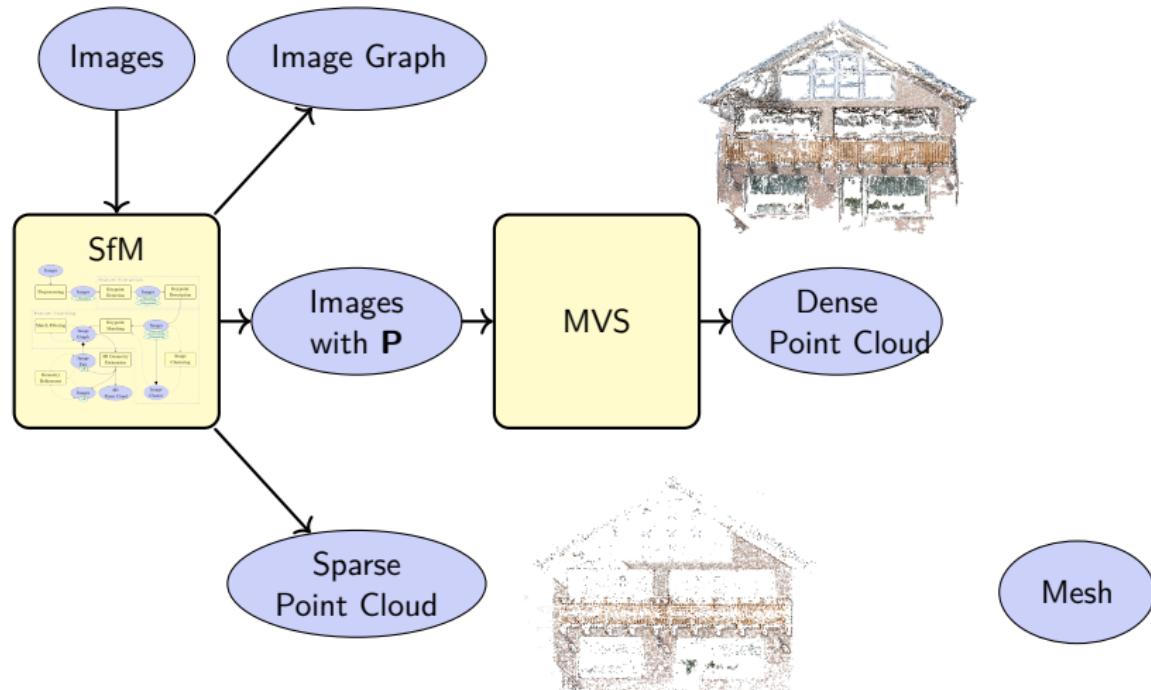
Highlevel overview of image-based 3D reconstruction



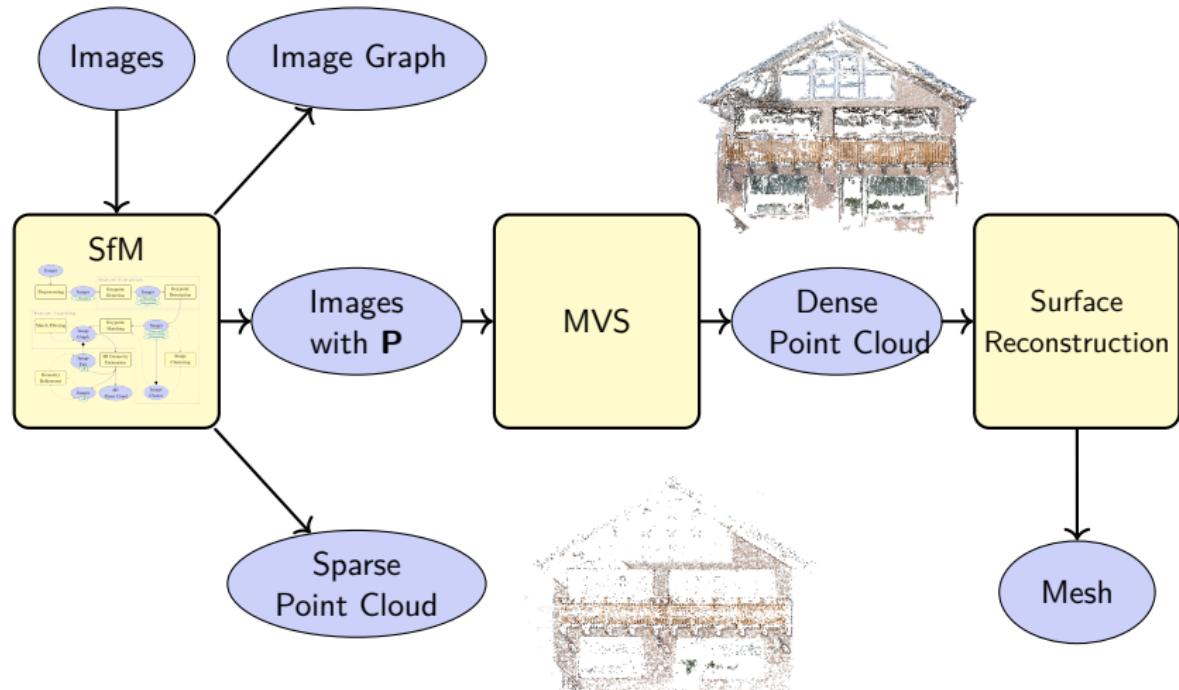
Highlevel overview of image-based 3D reconstruction



Highlevel overview of image-based 3D reconstruction



Highlevel overview of image-based 3D reconstruction



Highlevel overview of image-based 3D reconstruction

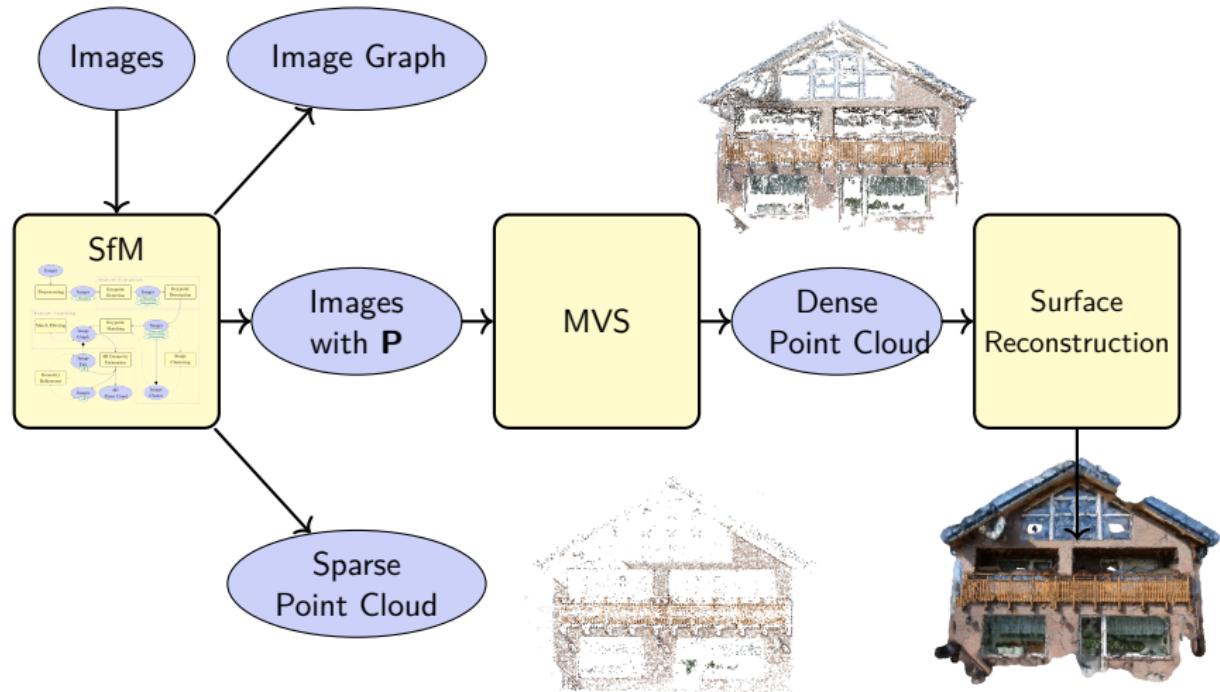


Table of Contents

- 1 Problem Definition
- 2 The Process of 3D Reconstruction
- 3 Framework Design and Implementation
- 4 Using the Framework for SfM

Module and Data

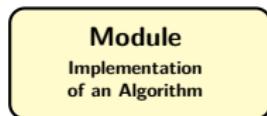
Module

Module
Implementation
of an Algorithm

Module and Data

Module

Data



Module and Data

Module

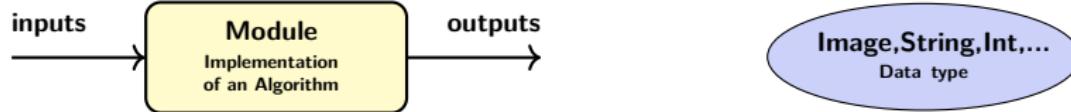
Data



Module and Data

Module

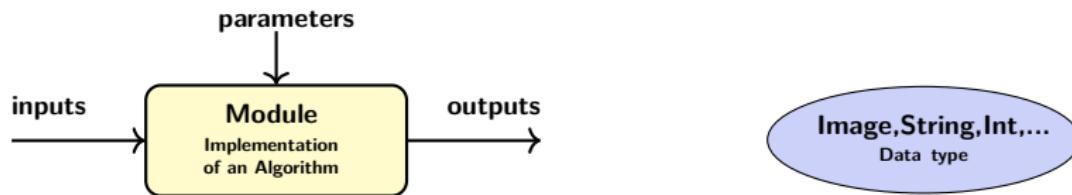
Data



Module and Data

Module

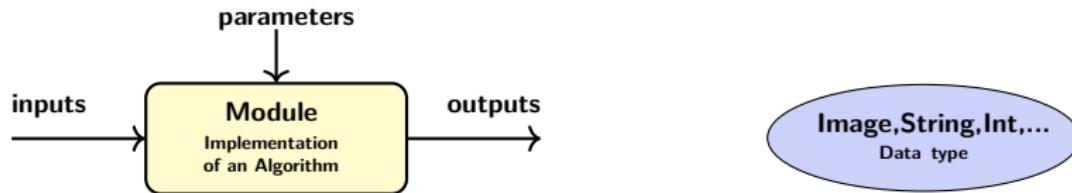
Data



Module and Data

Module

Data

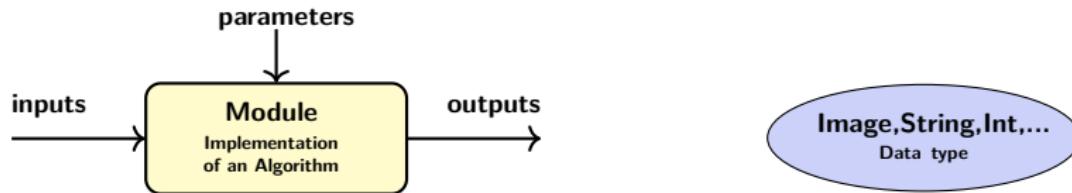


- clear interface → reuse of code in different contexts

Module and Data

Module

Data

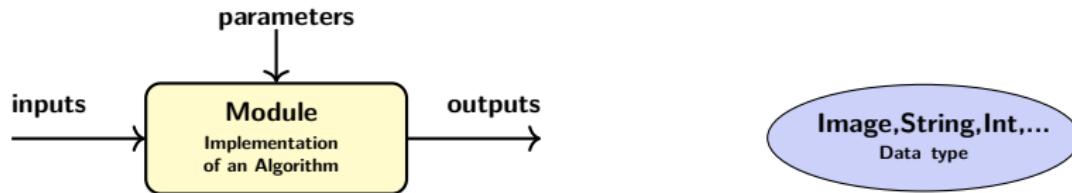


- clear interface → reuse of code in different contexts
- self-contained unit → parallelizable

Module and Data

Module

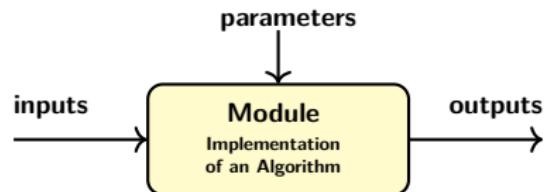
Data



- clear interface → reuse of code in different contexts
- self-contained unit → parallelizable
- common data type implementations → data flow control

Module and Data

Module

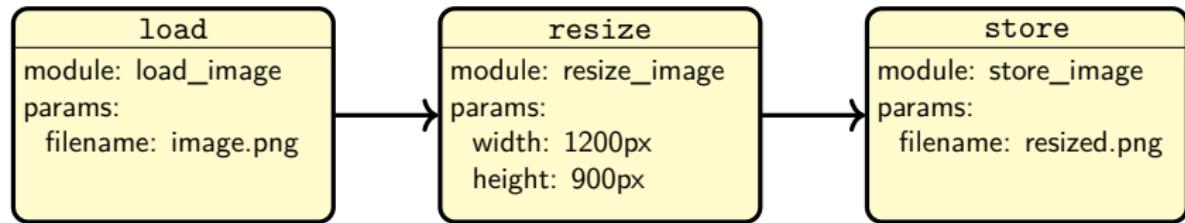


Data

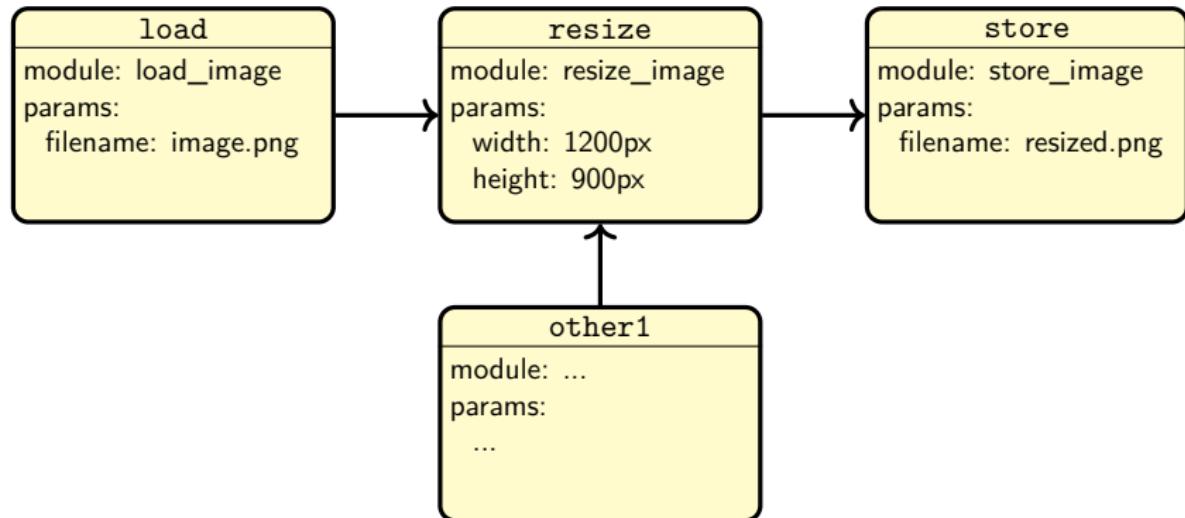


- clear interface → reuse of code in different contexts
- self-contained unit → parallelizable
- common data type implementations → data flow control
- processing chain is a dependency graph

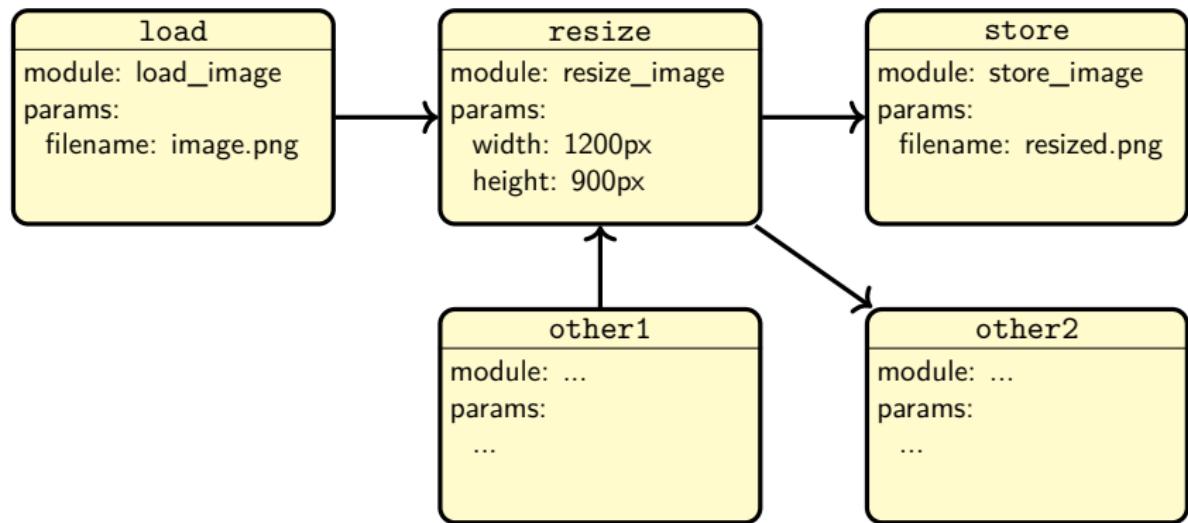
Processing Chain



Processing Chain



Processing Chain



Implementation of a Module

```
1 #include <uipf/data.hpp>
2 #include <uipf/data/opencv.hpp>
3
4 using namespace uipf;
5
6 #define UIPF_MODULE_ID "opencv.imgproc.resize"
7 #define UIPF_MODULE_NAME "Resize Image"
8 #define UIPF_MODULE_CATEGORY "opencv"
9 #define UIPF_MODULE_CLASS OpenCVResizeImage
// ...
```

Implementation of a Module

```
1 #include <uipf/data.hpp>
2 #include <uipf/data/opencv.hpp>
3
4 using namespace uipf;
5
6 #define UIPF_MODULE_ID "opencv.imgproc.resize"
7 #define UIPF_MODULE_NAME "Resize Image"
8 #define UIPF_MODULE_CATEGORY "opencv"
9 #define UIPF_MODULE_CLASS OpenCVResizeImage
// ...
```

Implementation of a Module

```
1 #include <uipf/data.hpp>
2 #include <uipf/data/opencv.hpp>
3
4 using namespace uipf;
5
6 #define UIPF_MODULE_ID "opencv.imgproc.resize"
7 #define UIPF_MODULE_NAME "Resize Image"
8 #define UIPF_MODULE_CATEGORY "opencv"
9 #define UIPF_MODULE_CLASS OpenCVResizeImage
// ...
```

Implementation of a Module

```
11 #define UIPF_MODULE_INPUTS \
12     {"image", DataDescription(data::OpenCVMat::id(), \
13                               "the input image.")}
14
15 #define UIPF_MODULE_OUTPUTS \
16     {"image", DataDescription(data::OpenCVMat::id(), \
17                               "the resized image.")}
18
19 #define UIPF_MODULE_PARAMS \
20     {"width", ParamDescription("new width."), \
21      {"height", ParamDescription("new height.")}}
22
23 #include <uipf/Module.hpp>
24
25 void OpenCVResizeImage::run() {
26     // Module Implementation goes here.
27 }
```

Implementation of a Module

```
11 #define UIPF_MODULE_INPUTS \
12     {"image", DataDescription(data::OpenCVMat::id(), \
13                               "the input image.")}
14
15 #define UIPF_MODULE_OUTPUTS \
16     {"image", DataDescription(data::OpenCVMat::id(), \
17                               "the resized image.")}
18
19 #define UIPF_MODULE_PARAMS \
20     {"width", ParamDescription("new width."), \
21      {"height", ParamDescription("new height.")}}
22
23 #include <uipf/Module.hpp>
24
25 void OpenCVResizeImage::run() {
26     // Module Implementation goes here.
27 }
```

Implementation of a Module

```
11 #define UIPF_MODULE_INPUTS \
12 { "image", DataDescription(data::OpenCVMat::id(), \
13                           "the input image.")}
14
15 #define UIPF_MODULE_OUTPUTS \
16 { "image", DataDescription(data::OpenCVMat::id(), \
17                           "the resized image.")}
18
19 #define UIPF_MODULE_PARAMS \
20 { "width", ParamDescription("new width."), \
21   {"height", ParamDescription("new height.")} }
22
23 #include <uipf/Module.hpp>
24
25 void OpenCVResizeImage::run() {
26   // Module Implementation goes here.
27 }
```

Implementation of a Module

```
11 #define UIPF_MODULE_INPUTS \
12 { "image", DataDescription(data::OpenCVMat::id(), \
13                           "the input image.")}
14
15 #define UIPF_MODULE_OUTPUTS \
16 { "image", DataDescription(data::OpenCVMat::id(), \
17                           "the resized image.")}
18
19 #define UIPF_MODULE_PARAMS \
20 { "width", ParamDescription("new width."), \
21   { "height", ParamDescription("new height.") } }
22
23 #include <uipf/Module.hpp>
24
25 void OpenCVResizeImage::run() {
26   // Module Implementation goes here.
27 }
```



Implementation of a Module

```
11 #define UIPF_MODULE_INPUTS \
12     {"image", DataDescription(data::OpenCVMat::id(), \
13                               "the input image.")}
14
15 #define UIPF_MODULE_OUTPUTS \
16     {"image", DataDescription(data::OpenCVMat::id(), \
17                               "the resized image.")}
18
19 #define UIPF_MODULE_PARAMS \
20     {"width", ParamDescription("new width."), \
21      {"height", ParamDescription("new height.")}}
22
23 #include <uipf/Module.hpp>
24
25 void OpenCVResizeImage::run() {
26     // Module Implementation goes here.
27 }
```

Implementation of a Module: Data Interface

```
1 void OpenCVResizeImage::run() {
2
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");
4     int width = getParam<int>("width", -1);
5     int height = getParam<int>("height", -1);
6
7     // do something here
8
9     OpenCVMat::ptr newImage(new OpenCVMat(m));
10    newImage->filename = image->filename;
11    newImage->exif = image->exif;
12    setOutputData<OpenCVMat>("image", newImage);
13
14 }
```

Implementation of a Module: Data Interface

```
1 void OpenCVResizeImage::run() {
2
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");
4     int width = getParam<int>("width", -1);
5     int height = getParam<int>("height", -1);
6
7     // do something here
8
9     OpenCVMat::ptr newImage(new OpenCVMat(m));
10    newImage->filename = image->filename;
11    newImage->exif = image->exif;
12    setOutputData<OpenCVMat>("image", newImage);
13
14 }
```

Implementation of a Module: Data Interface

```
1 void OpenCVResizeImage::run() {
2
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");
4     int width = getParam<int>("width", -1);
5     int height = getParam<int>("height", -1);
6
7     // do something here
8
9     OpenCVMat::ptr newImage(new OpenCVMat(m));
10    newImage->filename = image->filename;
11    newImage->exif = image->exif;
12    setOutputData<OpenCVMat>("image", newImage);
13
14 }
```

Implementation of a Module: Data Interface

```
1 void OpenCVResizeImage::run() {  
2  
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");  
4     int width = getParam<int>("width", -1);  
5     int height = getParam<int>("height", -1);  
6  
7     // do something here  
8  
9     OpenCVMat::ptr newImage(new OpenCVMat(m));  
10    newImage->filename = image->filename;  
11    newImage->exif = image->exif;  
12    setOutputData<OpenCVMat>("image", newImage);  
13  
14 }
```

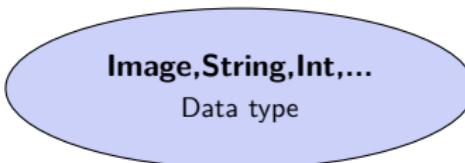
Implementation of a Module: Data Interface

```
1 void OpenCVResizeImage::run() {  
2  
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");  
4     int width = getParam<int>("width", -1);  
5     int height = getParam<int>("height", -1);  
6  
7     // do something here  
8  
9     OpenCVMat::ptr newImage(new OpenCVMat(m));  
10    newImage->filename = image->filename;  
11    newImage->exif = image->exif;  
12    setOutputData<OpenCVMat>("image", newImage);  
13  
14 }
```

Implementation of a Module: Data Interface

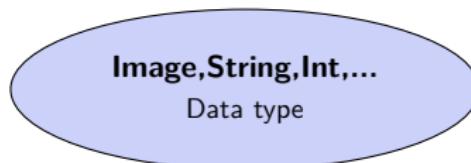
```
1 void OpenCVResizeImage::run() {  
2  
3     OpenCVMat::ptr img = getInputData<OpenCVMat>("image");  
4     int width = getParam<int>("width", -1);  
5     int height = getParam<int>("height", -1);  
6  
7     // do something here  
8  
9     OpenCVMat::ptr newImage(new OpenCVMat(m));  
10    newImage->filename = image->filename;  
11    newImage->exif = image->exif;  
12    setOutputData<OpenCVMat>("image", newImage);  
13  
14 }
```

Data structure implementation



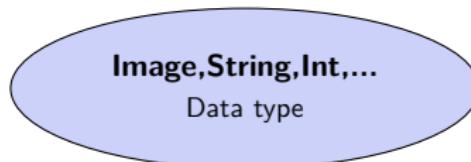
Image, String, Int, ...
Data type

Data structure implementation



- Data structure is a container

Data structure implementation



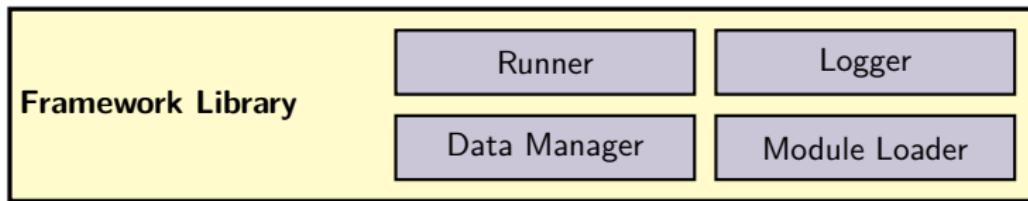
- Data structure is a container
- Interface for visualisation and serialisation

Data structure implementation

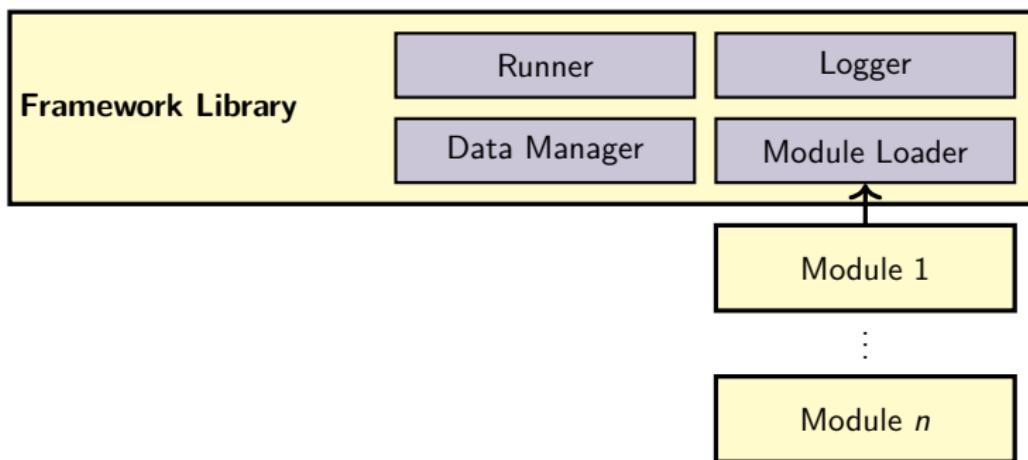


- Data structure is a container
- Interface for visualisation and serialisation
- Custom datatypes possible

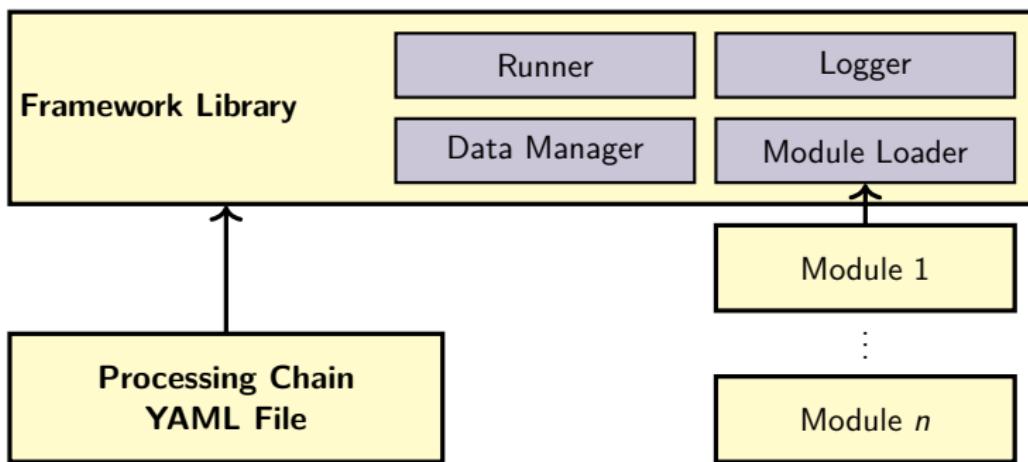
Framework overview



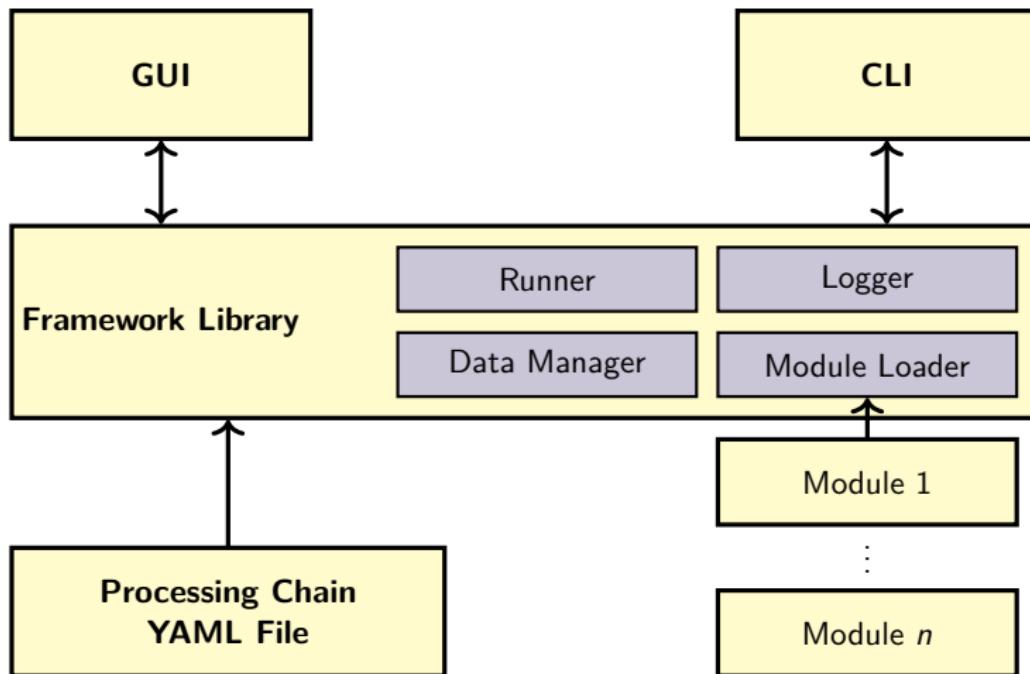
Framework overview



Framework overview



Framework overview



GU Interface

newFile.yaml - uipf

Processing chain:

- load images
- match
- pmvs

Add Processing Step **Delete Processing Step**

Step Configuration

Module: sfm cebe.sfm.p

Parameters

	Value
CPU	8
cszie	

Inputs from other steps:

	From Step:	Output Name:
imageGraph	SfM	imageGraph

Run Control

Resume Step Stop Clear

Chain: 57% (4/7)

Steps

- load images
- copy images
- SIFT
- match

Output Data

Output:	Type:	Serialize:	Visualize:
imageGraph	cebe.sfm....	n/a	items
			graph

Logs

Warn Error Info Filter Clear

```
deleted SIFT.image
on_workerDataDeleted: SIFTimage
match.imageGraph is requested.
```

```
graph TD; subgraph Chain [Processing chain]; direction TB; A[load images] --> B[match]; B --> C[pmvs]; end; subgraph Steps [Steps]; direction TB; D[SIFT] --> E[copy images]; E --> F[filter matches]; F --> G[SfM]; G --> H[pmvs]; end; subgraph Logs [Logs]; direction TB; I[deleted SIFT.image] --> J[on_workerDataDeleted: SIFTimage]; J --> K[match.imageGraph is requested.]; end;
```

CL Interface

```
# run a processing chain
uipf -c processing-chain.yaml
```

CL Interface

```
# run a processing chain
uipf -c processing-chain.yaml
```

```
# run a single module
uipf uipfsfm.keypoint -i image.jpg -o points.txt
```

CL Interface

```
# run a processing chain
uipf -c processing-chain.yaml

# run a single module
uipf uipfsfm.keypoint -i image.jpg -o points.txt

# list and document modules
uipf -l
uipf --info uipfsfm.keypoint.sift
```

Table of Contents

- 1 Problem Definition
- 2 The Process of 3D Reconstruction
- 3 Framework Design and Implementation
- 4 Using the Framework for SfM

Dataset



79 images
2592 × 1728px

"Der Hass" dataset by [Fuhrmann 2014]

Simon Fuhrmann, Fabian Langguth and Michael Goesele. MVE - A Multi-View Reconstruction Environment.
In GCH, pages 1118, 2014.

Processing Chain

/home/cebe/Dokumente/Uni/master/Experiments/der_hass/2_sfms_local.yml - uipf

File Edit Configuration View Help

Processing chain:
bundler_sfm
 copy images
 loadImages

Add Processing Step Delete Processing Step

Step Configuration
Module: sfm **cebe.sfm.b**

Parameters

	Value
workdir	tmp_sfm

Inputs from other steps:

	From Step:	Output Name:
imageGraph	bundler_mat...	imageGraph

Run Control
 Resume Step... Stop Clear
 Chain: 37% (5/11) Module: 12%

Steps

loadImages
copy images
resizeAll
to_sfm
sift

Output Data
 Output: Type: Serialize: Visualize:

Logs

Warn Error Info Filter

```
Number of detected interest points:./tmp/images/IMG_0165.JPG: 544
using cache: 1
creating temporary .pgm file for image ./tmp/images/IMG_0207.JPG.pgm
```

Processing Chain

The screenshot shows the Processing Chain application window with the following details:

- Processing chain:** A list on the left shows the current chain: `bundler_sfm` (highlighted), `copy images`, and `loadImages`. Buttons for `Add Processing Step` and `Delete Processing Step` are available.
- Run Control:** A panel on the right displays the progress of the current step: "Chain: 37% (5/11)" and "Module: 12%". It includes buttons for `Resume`, `Step...`, `Stop`, and `Clear`.
- Step Configuration:** A large central panel shows the configuration for the selected step, `bundler_matcher`. It lists items in the output 'image' of the step 'sift'. The table has columns for Item, Type, Visualize, and steps 4 through 7.

Item:	Type:	Visualize:	4	5	6	7
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
- Logs:** A panel on the right shows logs for the current step. It includes buttons for `Clear` and `Save`.
- Configuration Buttons:** At the bottom right are buttons for `Save`, `Load`, `Cancel`, and `OK`.

Processing Chain

Processing chain:

```
bundler_sfm
copy images
loadImages
```

Add Processing Step **Delete Processing Step**

Step Configuration

Module	Item:	Type:	Visualize:	...			
Parameter	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Parameter	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Parameter	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Parameter	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Parameter	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Inputs	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Inputs	./tmp/images/l...	cebe.sfm.data...	image	keypoint			
Inputs	./tmp/images/l...	cebe.sfm.data...	image	keypoints			
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF
Logs	./tmp/images/l...	cebe.sfm.data...	image	keypoints	keypoints.txt	params	EXIF

Logs

Write logs using **create**

Image

Visualize:

Clear

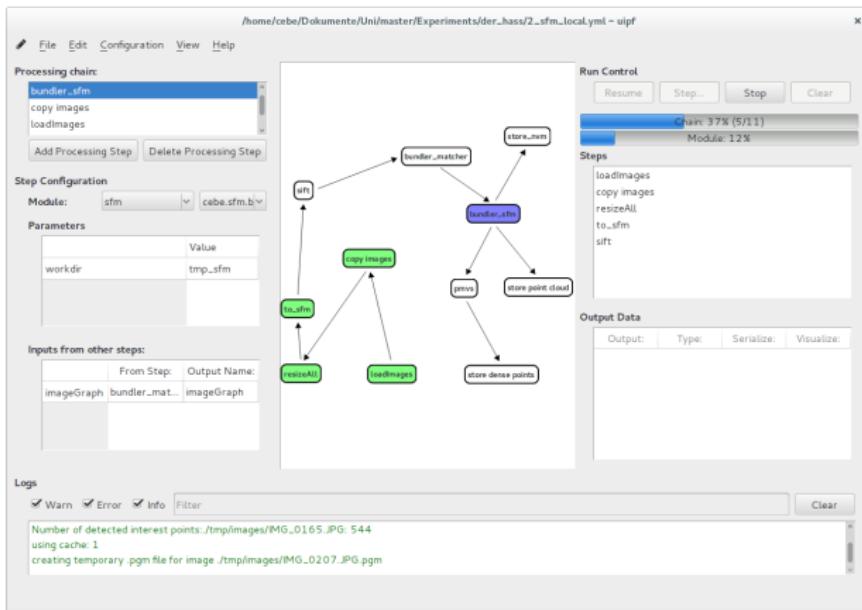
Sparse Point Cloud



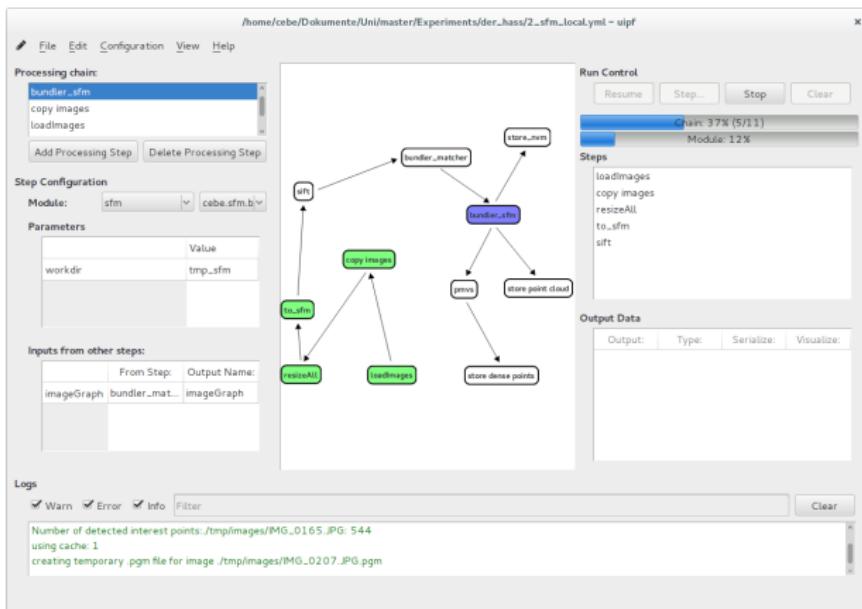
Mesh



Conclusion



Conclusion



<https://github.com/uipf/uipf>

