



WESTFÄLISCHE
WILHELMS-UNIVERSITÄT
MÜNSTER

Masterthesis

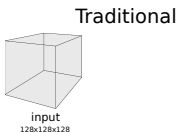
Efficient Implementation and Optimization of Geometric Multigrid Operations in
the LIFT Framework

Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)

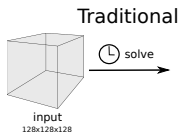
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)



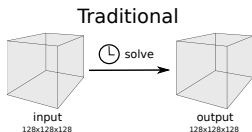
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)



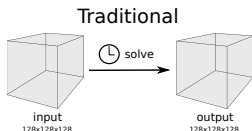
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)



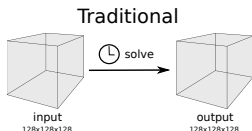
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



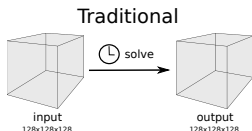
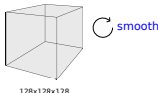
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



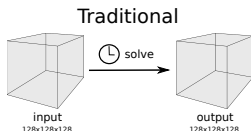
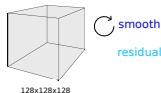
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



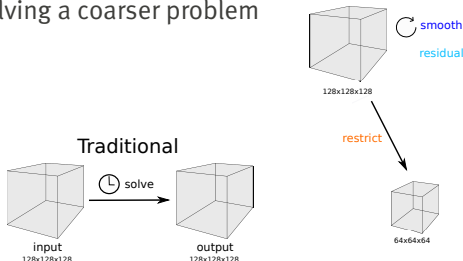
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



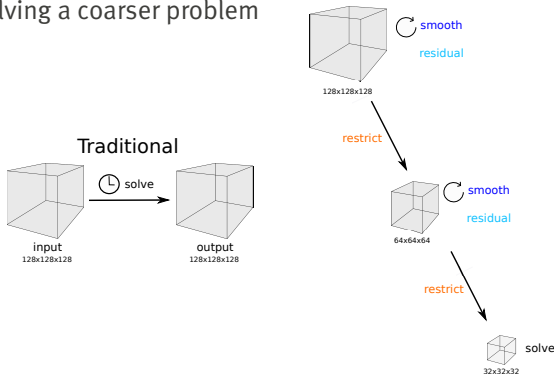
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



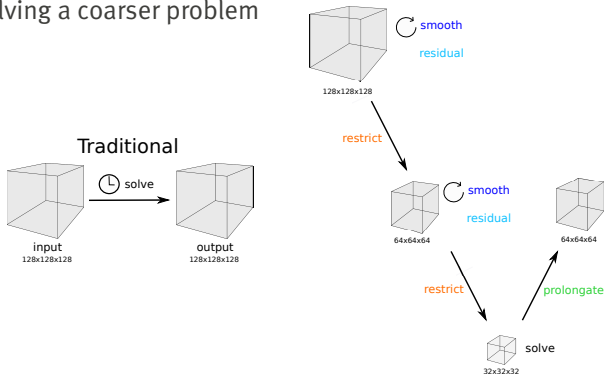
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



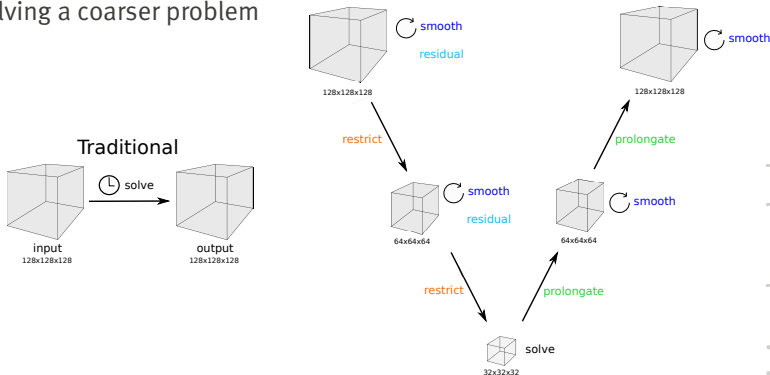
Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem



Geometric Multigrid

- ▶ Class of Iterative Solvers for Partial Differential Equations (PDE)
- ▶ Idea: Correction of the fine grid solution approximation by solving a coarser problem





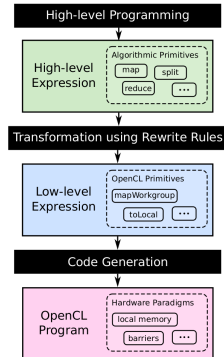
WESTFÄLISCHE
WILHELMS-UNIVERSITÄT
MÜNSTER

LIFT

Masterthesis | 3

LIFT

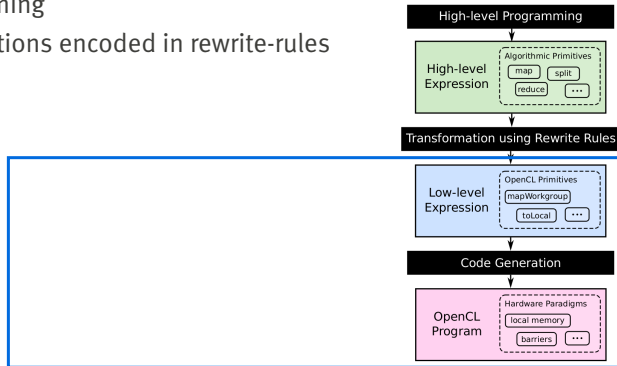
- ▶ Generating high-performance programs from functional programming
- ▶ Optimizations encoded in rewrite-rules



LIFT

- ▶ Generating high-performance programs from functional programming
- ▶ Optimizations encoded in rewrite-rules

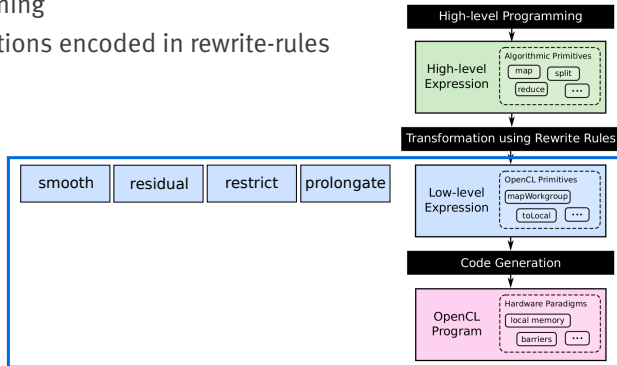
This Thesis:



LIFT

- ▶ Generating high-performance programs from functional programming
- ▶ Optimizations encoded in rewrite-rules

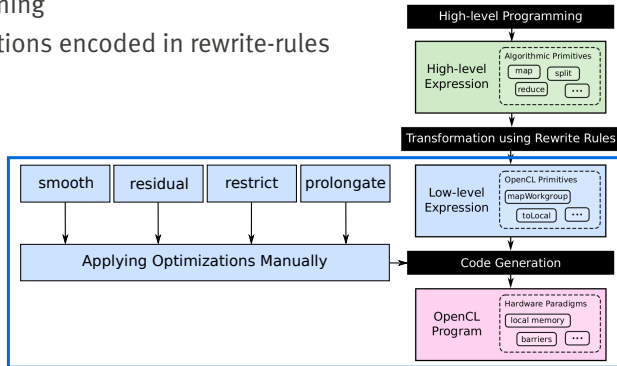
This Thesis:



LIFT

- ▶ Generating high-performance programs from functional programming
- ▶ Optimizations encoded in rewrite-rules

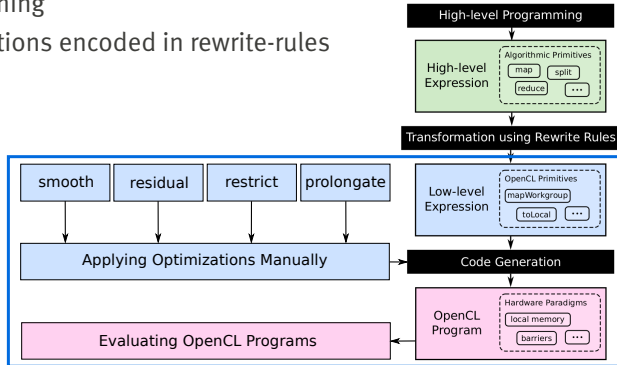
This Thesis:



LIFT

- ▶ Generating high-performance programs from functional programming
- ▶ Optimizations encoded in rewrite-rules

This Thesis:



Designing LIFT Expressions

- ▶ Identifying the general structure of the operation

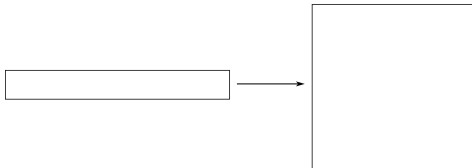
Designing LIFT Expressions

- ▶ Identifying the general structure of the operation
- ▶ Expressing the operation in 1 dimension



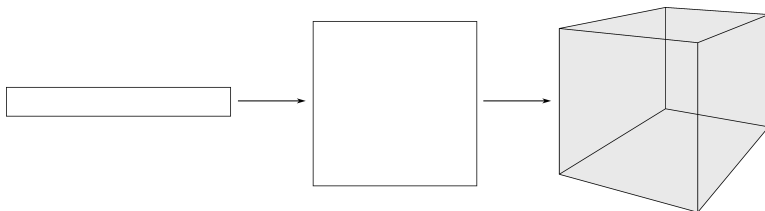
Designing LIFT Expressions

- ▶ Identifying the general structure of the operation
- ▶ Expressing the operation in 1 dimension
- ▶ Scaling the operation to 2 and 3 dimensions

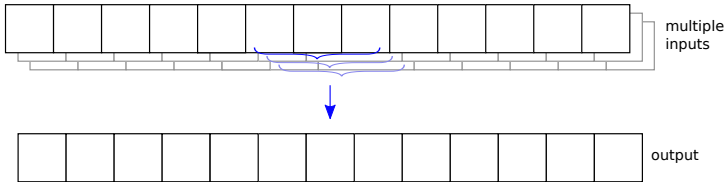
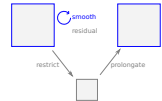


Designing LIFT Expressions

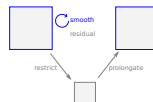
- ▶ Identifying the general structure of the operation
- ▶ Expressing the operation in 1 dimension
- ▶ Scaling the operation to 2 and 3 dimensions



Smooth & Residual

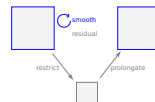
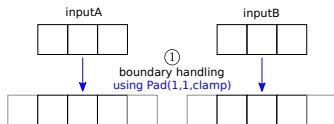


Smooth & Residual



A,
B

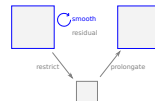
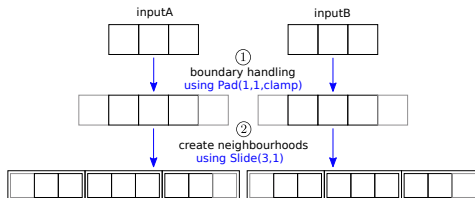
Smooth & Residual



Pad(1,1,clamp) \$ A,

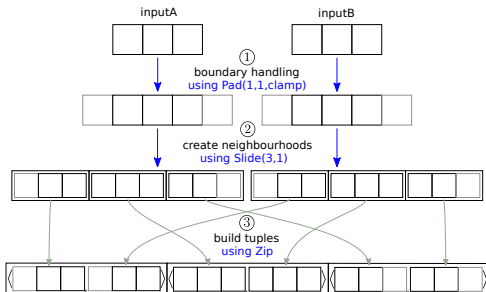
Pad(1,1,clamp) \$ B

Smooth & Residual

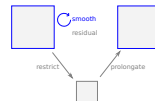


Slide(3,1) o Pad(1,1,clamp) \$ A,
Slide(3,1) o Pad(1,1,clamp) \$ B

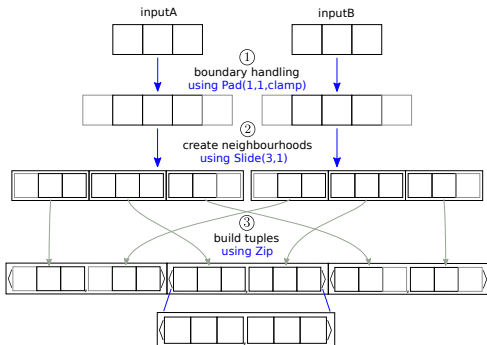
Smooth & Residual



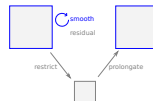
```
Zip(
  Slide(3,1) o Pad(1,1,clamp) $ A,
  Slide(3,1) o Pad(1,1,clamp) $ B )
```



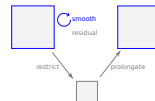
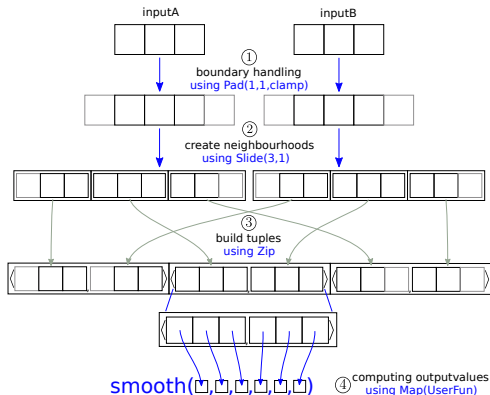
Smooth & Residual



```
MapGlb( compute ) o
Zip(
  Slide(3,1) o Pad(1,1,clamp) $ A,
  Slide(3,1) o Pad(1,1,clamp) $ B )
```



Smooth & Residual

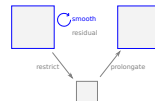
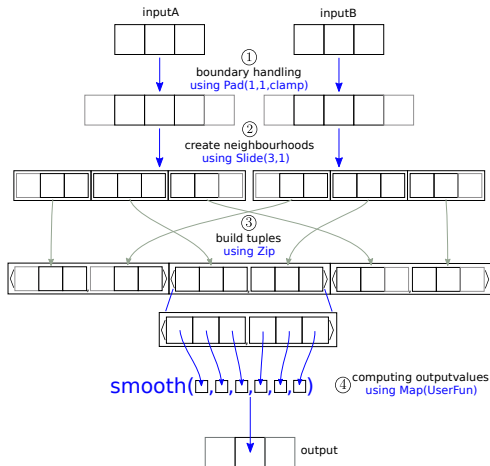


```
compute(Tuple t) = {
  a1 = tuple<0>[0]
  a2 = tuple<0>[1]
  a3 = tuple<0>[2]
  b1 = tuple<1>[0]
  b2 = tuple<1>[1]
  b3 = tuple<1>[2]

  smooth(a1, a2, a3, b1, b2, b3)
}

MapGlb( compute ) o
Zip(
  Slide(3,1) o Pad(1,1,clamp) $ A,
  Slide(3,1) o Pad(1,1,clamp) $ B )
```

Smooth & Residual

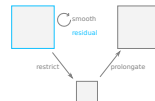
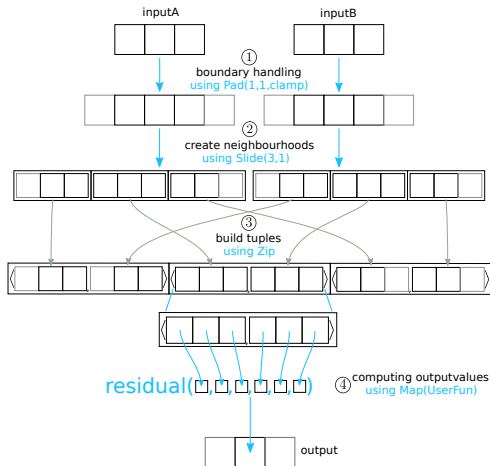


```
compute(Tuple t) = {
  a1 = tuple<0>[0]
  a2 = tuple<0>[1]
  a3 = tuple<0>[2]
  b1 = tuple<1>[0]
  b2 = tuple<1>[1]
  b3 = tuple<1>[2]

  smooth(a1, a2, a3, b1, b2, b3)
}

MapGlb( compute ) o
Zip(
  Slide(3,1) o Pad(1,1,clamp) $ A,
  Slide(3,1) o Pad(1,1,clamp) $ B )
```

Smooth & Residual

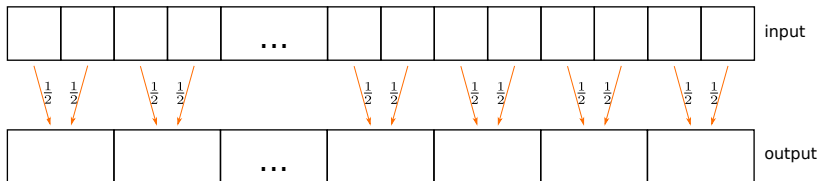
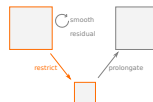


```
compute(Tuple t) = {
  a1 = tuple<0>[0]
  a2 = tuple<0>[1]
  a3 = tuple<0>[2]
  b1 = tuple<1>[0]
  b2 = tuple<1>[1]
  b3 = tuple<1>[2]

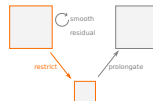
  residual(a1, a2, a3, b1, b2, b3)
}
```

```
MapGlb( compute ) o
Zip(
  Slide(3,1) o Pad(1,1,clamp) $ A,
  Slide(3,1) o Pad(1,1,clamp) $ B )
```


Restriction 1D

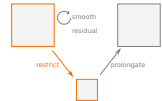
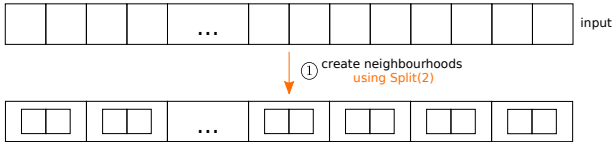


Restriction 1D



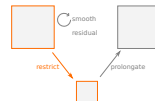
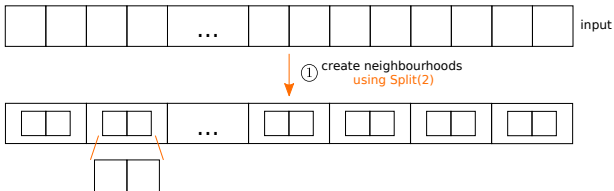
input

Restriction 1D



Split(2) \$ input

Restriction 1D

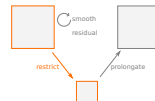
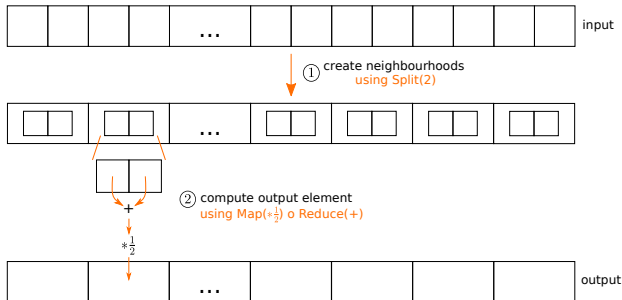


MapGlb(

) o

Split(2) \$ input

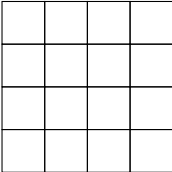
Restriction 1D



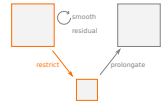
```
MapGlb(  
  Map( $\ast \frac{1}{2}$ ) o  
  Reduce(+)  
) o  
Split(2) $ input
```

Restriction 2D

input

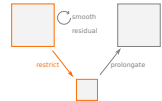
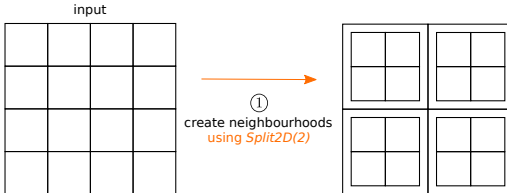


Masterthesis 7



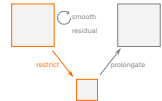
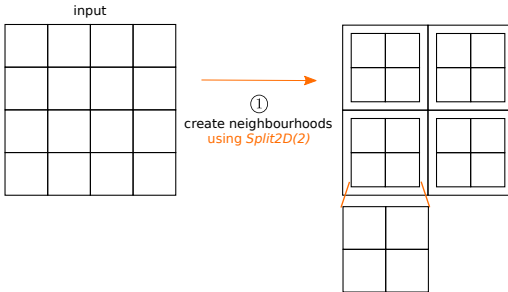
input

Restriction 2D



Split2D(2) \$ input

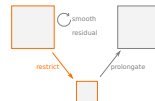
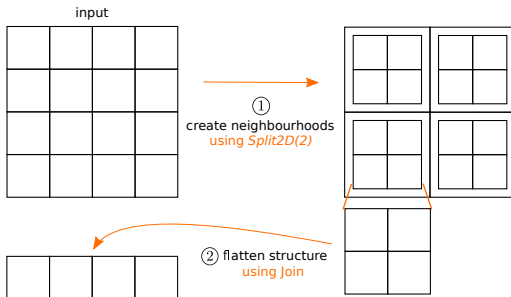
Restriction 2D



```
MapGlb(  
  MapGlb(  
    
```

```
  )) o  
  Split2D(2) $ input
```

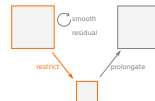
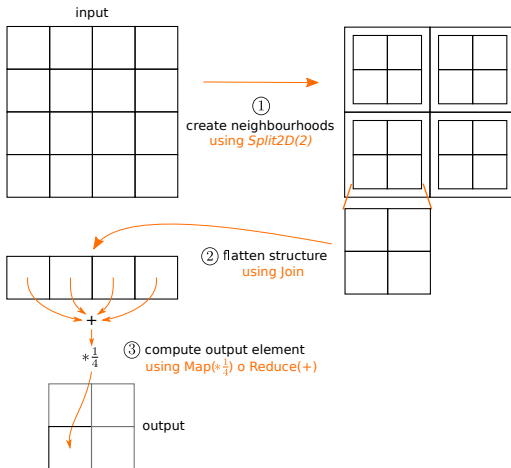

Restriction 2D



```
MapGlb(
  MapGlb(

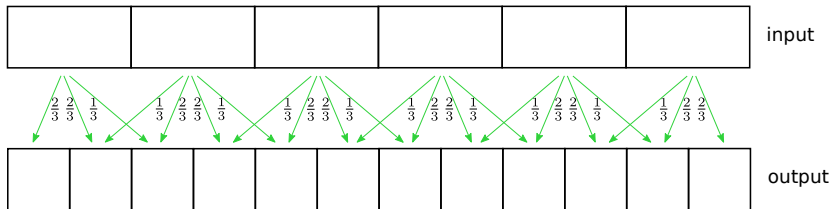
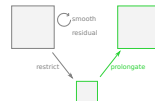
    Join
  )) o
  Split2D(2) $ input
```

Restriction 2D

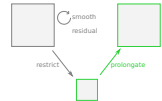
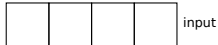


```
MapGlb(
  MapGlb(
    Map( $\ast \frac{1}{4}$ ) o
    Reduce(+) o
    Join
  ) o
  Split2D(2) $ input
```

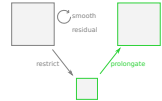
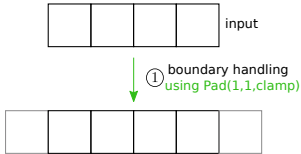
Prolongation



Prolongation

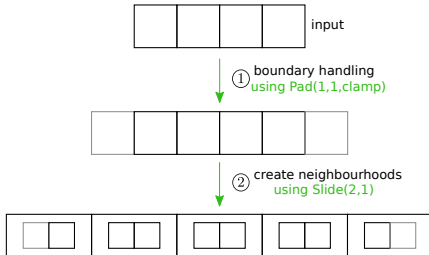


Prolongation

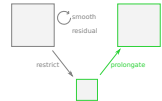


$\text{Pad}(1,1,\text{clamp})$ \$ input

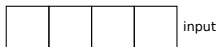
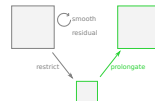
Prolongation



Slide(2,1) o
Pad(1,1,clamp) \$ input



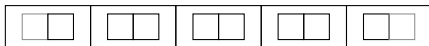
Prolongation



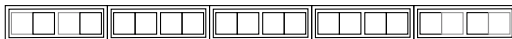
① boundary handling
using $\text{Pad}(1,1,\text{clamp})$



② create neighbourhoods
using $\text{Slide}(2,1)$

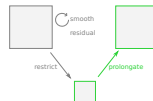
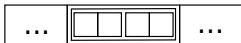


③ duplicate neighbourhoods
using $\text{Map}(\text{Pad}(1,0,\text{clamp})) \circ \text{Split}(1)$



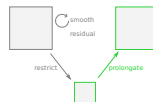
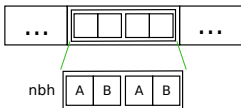
$\text{Map}(\text{Pad}(1,0,\text{clamp})) \circ$
 $\text{Split}(1) \circ$
 $\text{Slide}(2,1) \circ$
 $\text{Pad}(1,1,\text{clamp}) \circ \text{input}$

Prolongation



```
Map(Pad(1,0,wrap)) o
Split(1) o
Slide(2,1) o
Pad(1,1,clamp) $ input
```


Prolongation



Map(nbh =>

nbh,

) o

Map(Pad(1,0,wrap)) o

Split(1) o

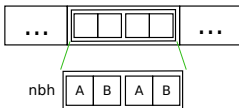
Slide(2,1) o

Pad(1,1,clamp) \$ input

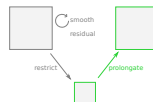
Prolongation

weights

x	y
---	---

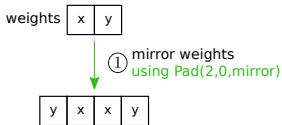
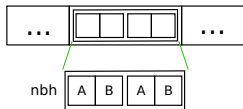


Masterthesis 9

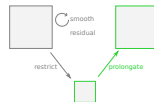


```
Map(nbh =>
    nbh,
    weights
) o
Map(Pad(1,0,wrap)) o
Split(1) o
Slide(2,1) o
Pad(1,1,clamp) $ input
```

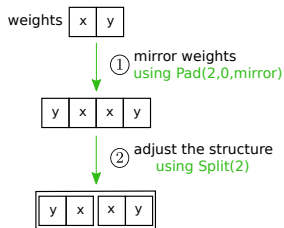
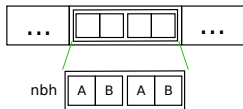
Prolongation



```
Map(nbh =>
    nbh,
    weights
    Pad(2,0,Mirror) $
) o
Map(Pad(1,0,wrap)) o
Split(1) o
Slide(2,1) o
Pad(1,1,clamp) $ input
```



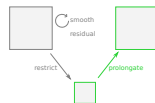
Prolongation



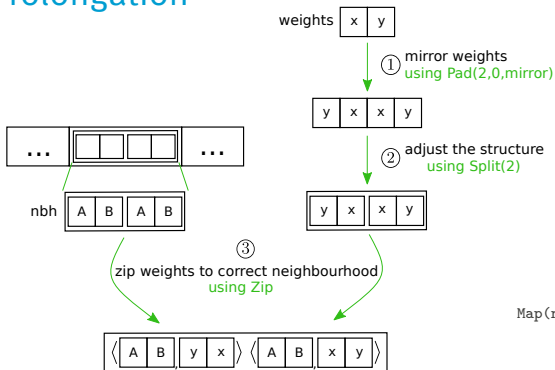
Map(nbh =>

nbh,
Split(2) o Pad(2,0,Mirror) \$
weights

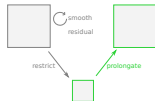
) o
Map(Pad(1,0,wrap)) o
Split(1) o
Slide(2,1) o
Pad(1,1,clamp) \$ input



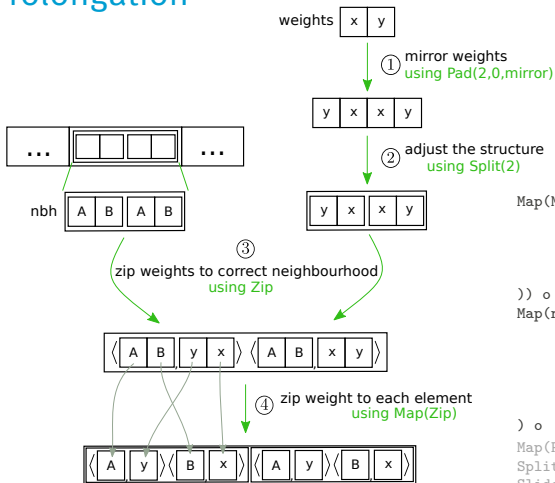
Prolongation



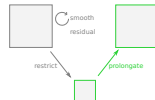
```
Map(nbh =>
  Zip(
    nbh,
    Split(2) o Pad(2,0,Mirror) $
    weights
  ) o
  Map(Pad(1,0,wrap)) o
  Split(1) o
  Slide(2,1) o
  Pad(1,1,clamp) $ input
```



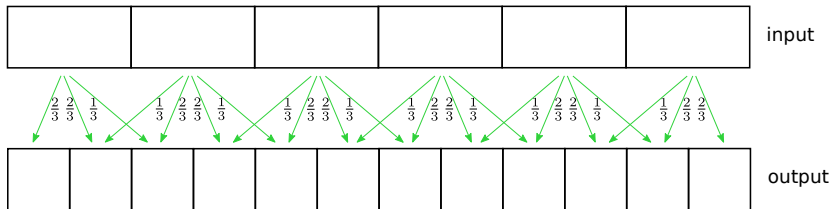
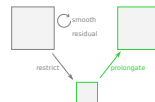
Prolongation



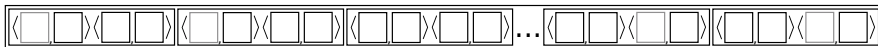
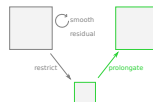
```
Map(Map(tuple =>
  Zip(
    tuple<0>,
    tuple<1>
  )
)) o
Map(nbh =>
  Zip(
    nbh,
    Split(2) o Pad(2,0,Mirror) $
    weights
  )
) o
Map(Pad(1,0,wrap)) o
Split(1) o
Slide(2,1) o
Pad(1,1,clamp) $ input
```



Prolongation

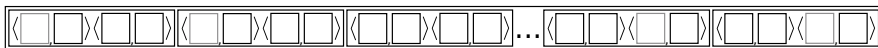
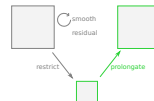


Prolongation

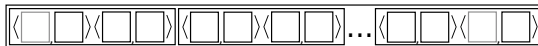


```
Map(Map(tuple =>
  Zip(
    tuple<0>,
    tuple<1>)
  )) o
[...]
```


Prolongation

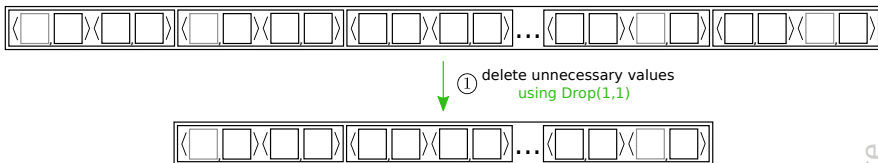
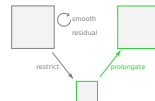


① delete unnecessary values
using `Drop(1,1)`



```
Drop(1,1) o
Map(Map(tuple =>
  Zip(
    tuple<0>,
    tuple<1>)
  )) o
[...]
```

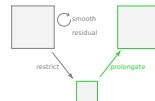
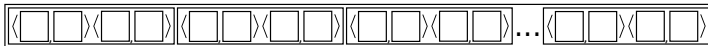
Prolongation



- ▶ $\text{Drop}(1, r)$ was introduced in this thesis
- ▶ Extends LIFT to be capable of expressing prolongation

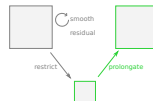
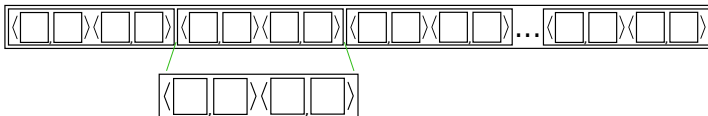
```
Drop(1,1) o
Map(Map(tuple =>
  Zip(
    tuple<0>,
    tuple<1>)
  )) o
[...]
```

Prolongation



$\text{Drop}(1,1) \circ$
[...]

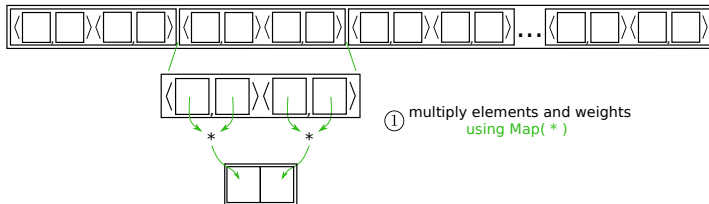
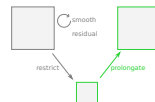
Prolongation



MapGlb(

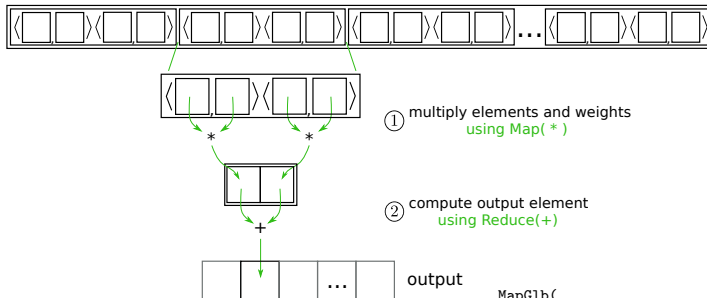
) o
Drop(1,1) o
[...]

Prolongation

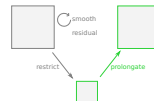


```
MapGlb(  
    MapSeq(  
        tuple<0> * tuple<1>  
    )  
    ) o  
Drop(1,1) o  
[...]
```

Prolongation



```
MapGlb(
  Reduce(+) o
  MapSeq(
    tuple<0> * tuple<1>
  )
) o
Drop(1,1) o
[...]
```



Evaluation

- ▶ LIFT does not support building entire programs yet
- ▶ All operations expressed in LIFT in multiple dimensions

Evaluation

- ▶ LIFT does not support building entire programs yet
- ▶ All operations expressed in LIFT in multiple dimensions

Performance comparison of individual operations

- ▶ OpenCL kernels generated from handwritten LIFT low-level expressions
- ▶ OpenCL kernels generated with the Polyhedral Parallel Code Generation (PPCG) compiler

Evaluation

- ▶ LIFT does not support building entire programs yet
- ▶ All operations expressed in LIFT in multiple dimensions

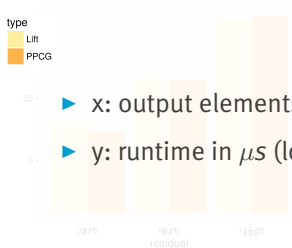
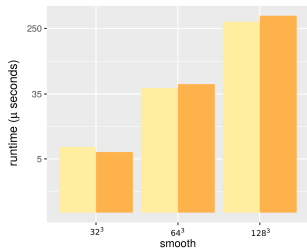
Performance comparison of individual operations

- ▶ OpenCL kernels generated from handwritten LIFT low-level expressions
- ▶ OpenCL kernels generated with the Polyhedral Parallel Code Generation (PPCG) compiler

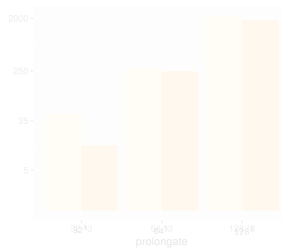
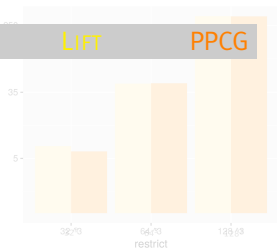
Workflow

- ▶ Used GPU: Nvidia GeForce GTX 1080
- ▶ All parameters auto-tuned with the Auto Tuning Framework (ATF)
- ▶ Tuning time: 1 hour per framework per operation per input size

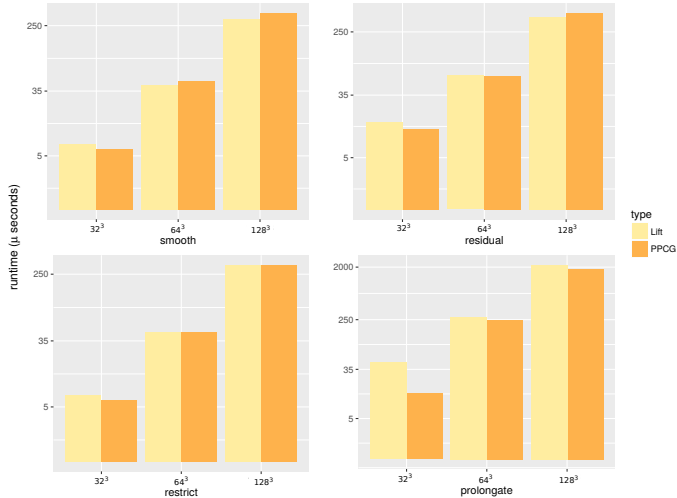
Runtime Comparison



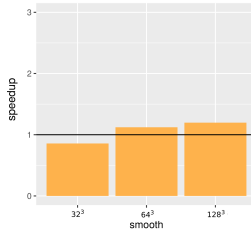
- ▶ x: output elements (32^3 , 64^3 , 128^3)
- ▶ y: runtime in μs (lower is better)



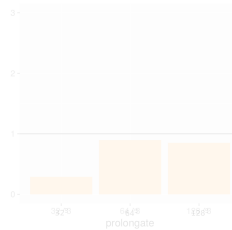
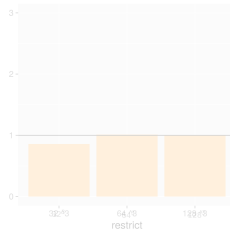
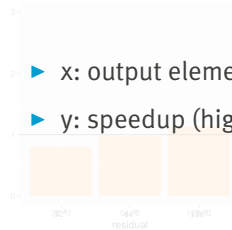
Runtime Comparison



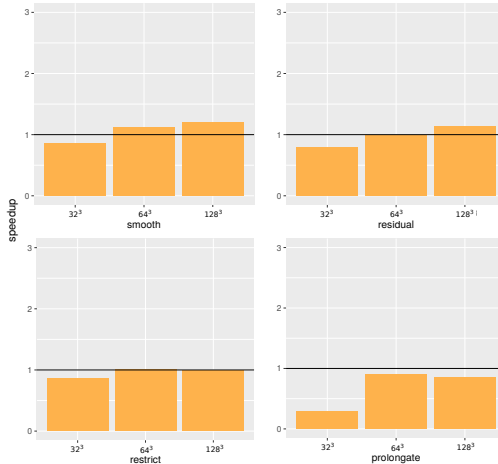
Speedup Comparison



- ▶ x: output elements (32^3 , 64^3 , 128^3)
- ▶ y: speedup (higher is better)



Speedup Comparison

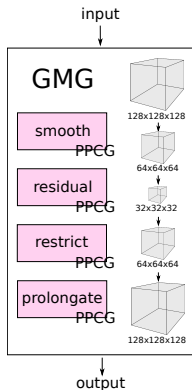


GMG Program

In a GMG solver the operations vary in number of execution, so further evaluation was necessary:

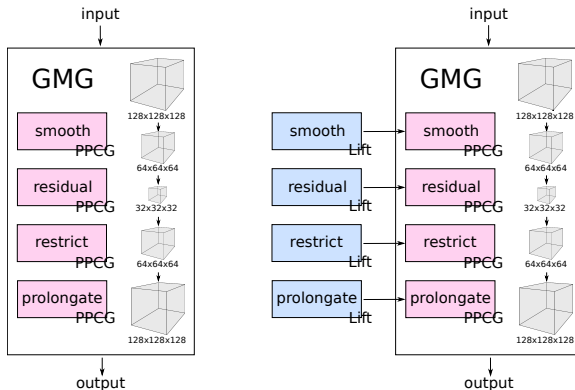
GMG Program

In a GMG solver the operations vary in number of execution, so further evaluation was necessary:



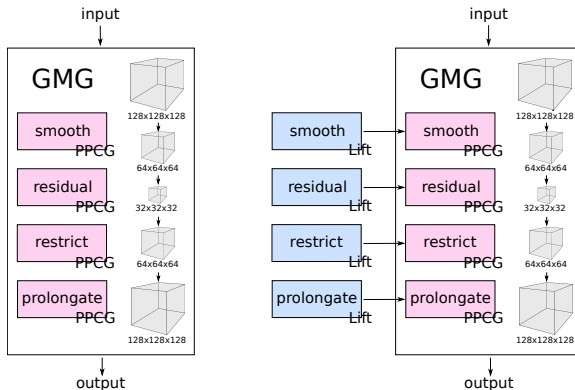
GMG Program

In a GMG solver the operations vary in number of execution, so further evaluation was necessary:



GMG Program

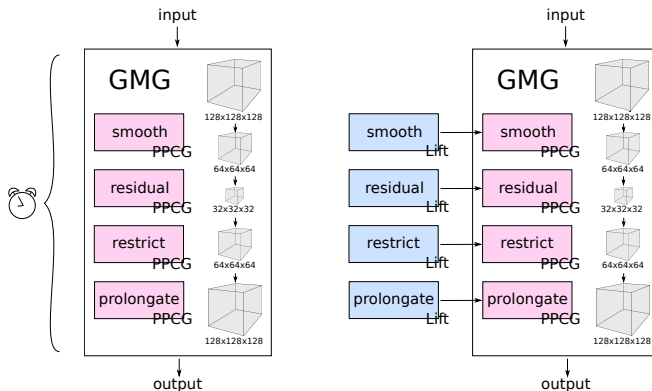
In a GMG solver the operations vary in number of execution, so further evaluation was necessary:



For each operation the best parameters from the previous auto-tuning are used

GMG Program

In a GMG solver the operations vary in number of execution, so further evaluation was necessary:



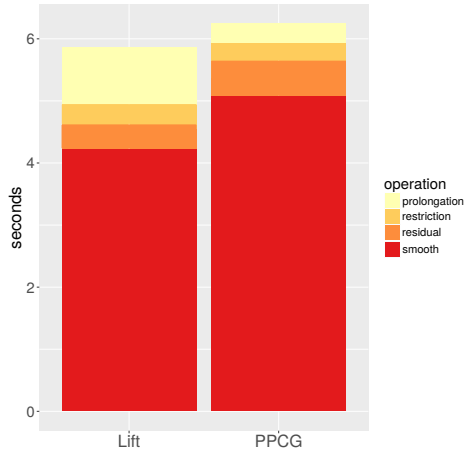
For each operation the best parameters from the previous auto-tuning are used

GMG Program Runtime Comparison

- ▶ Residual, restrict, prolongate each executed 2 times
- ▶ Smooth executed 24 times

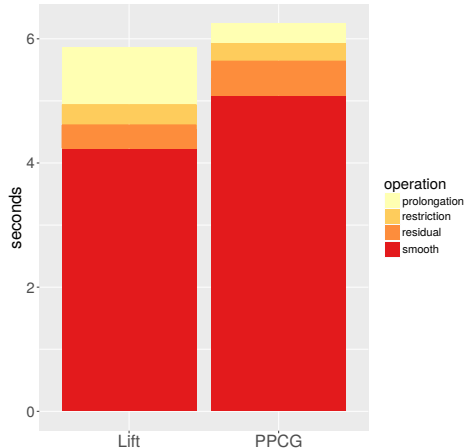
GMG Program Runtime Comparison

- ▶ Residual, restrict, prolongate each executed 2 times
- ▶ Smooth executed 24 times



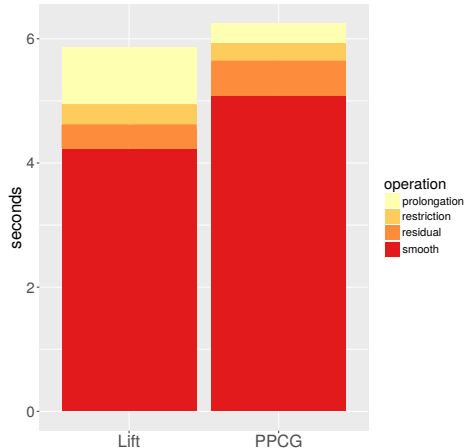
GMG Program Runtime Comparison

- ▶ Residual, restrict, prolongate each executed 2 times
- ▶ Smooth executed 24 times
- ▶ Small improvement in smooth has large impact on overall runtime



GMG Program Runtime Comparison

- ▶ Residual, restrict, prolongate each executed 2 times
- ▶ Smooth executed 24 times
- ▶ Small improvement in smooth has large impact on overall runtime
- ▶ Experiments with optimizations for iterative kernels in LIFT



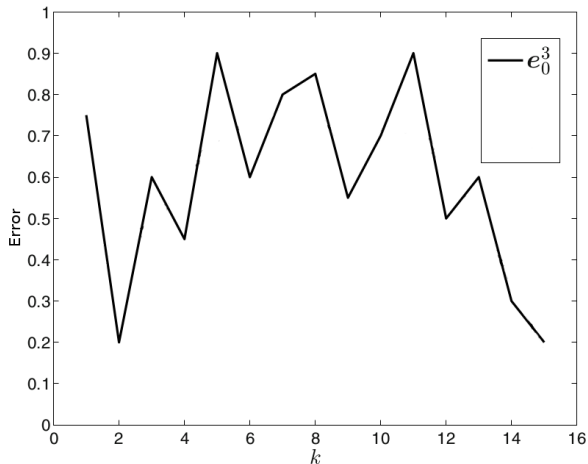


Questions?

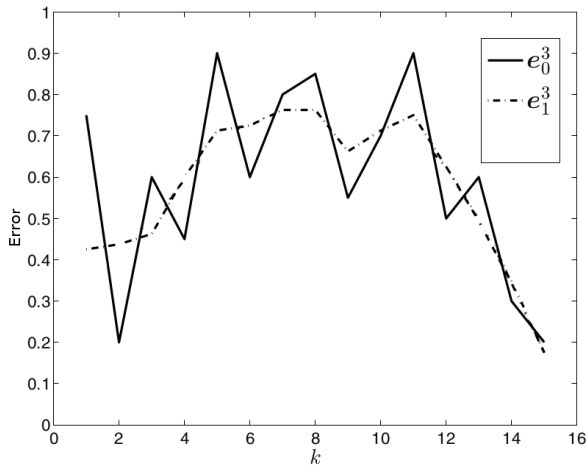


Backup Slides

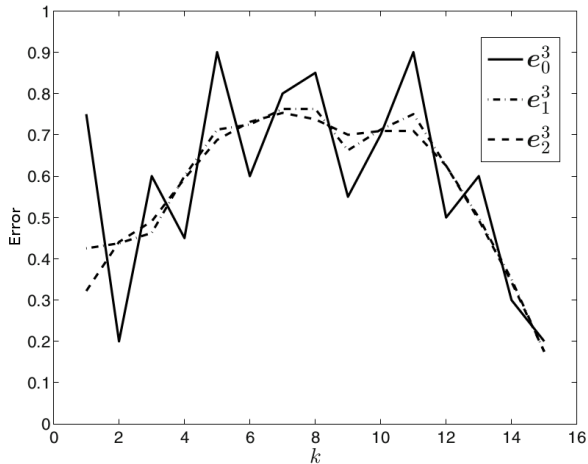
Error Smoothing



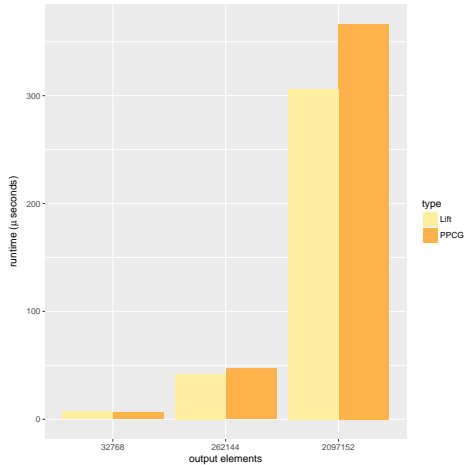
Error Smoothing



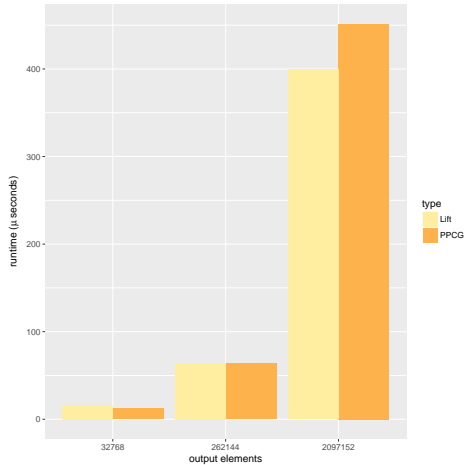
Error Smoothing



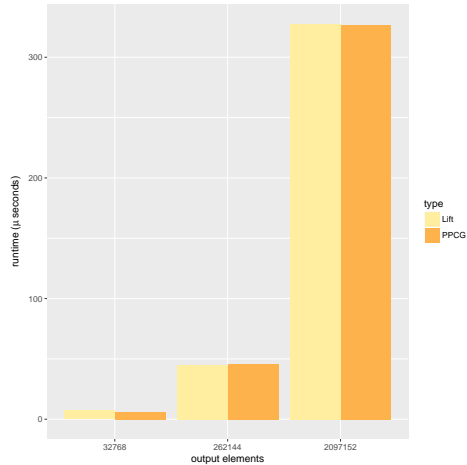
Smooth



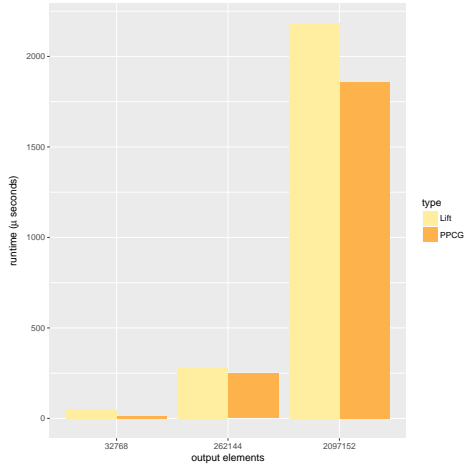
Residual



Restrict



Interpolate



Comparison

