Exploiting Temporal and Spatial Coherence in Hierarchical Visibility Algorithms

Jiří Bittner and Vlastimil Havran
Centre of Applied Cybernetics
Czech Technical University in Prague

- **Introduction, hierarchical visibility.**
- **Hierarchy updating, conservative hierarchy updating.**
- **Visibility propagation.**
- **Results, conclusion, future work.**

Hierarchical Visibility Algorithms

Computer Graphics Group

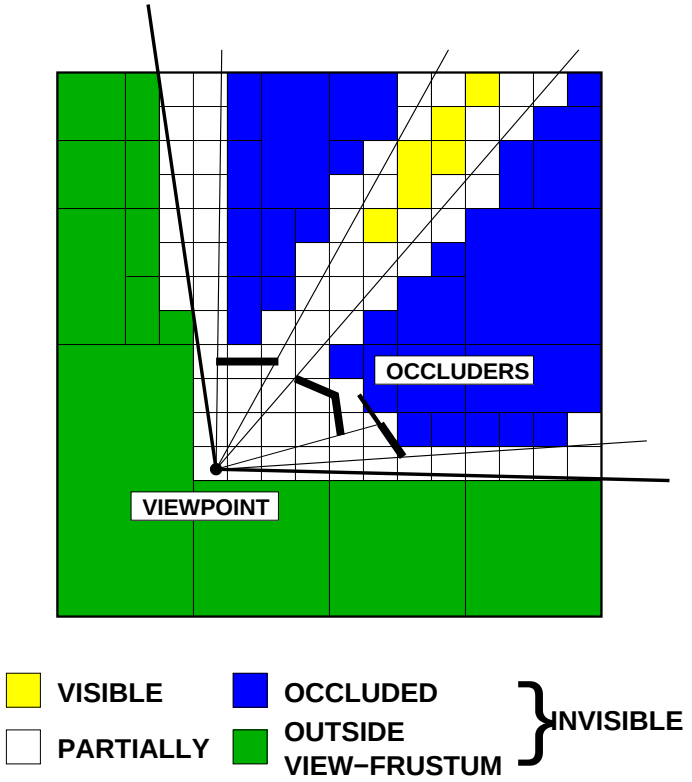
Speedup rendering of large scenes by visibility culling

Spatial hierarchy

- **kD–tree**, octree, OBB tree

Visibility test of a region:

- visible (V)
- invisible (I)
- partially visible (PV)



- for **each viewpoint** start at the root of the hierarchy
- recursivelly apply visibility test on **PV** regions

Hierarchical Visibility Algorithms

Computer Graphics Group

Image space

- Hierarchical z–buffer. Greene ’93.
- Hierarchical occlusion maps. Manocha et al. ’96.
- Hierarchical polygon tiling. Greene ’96.

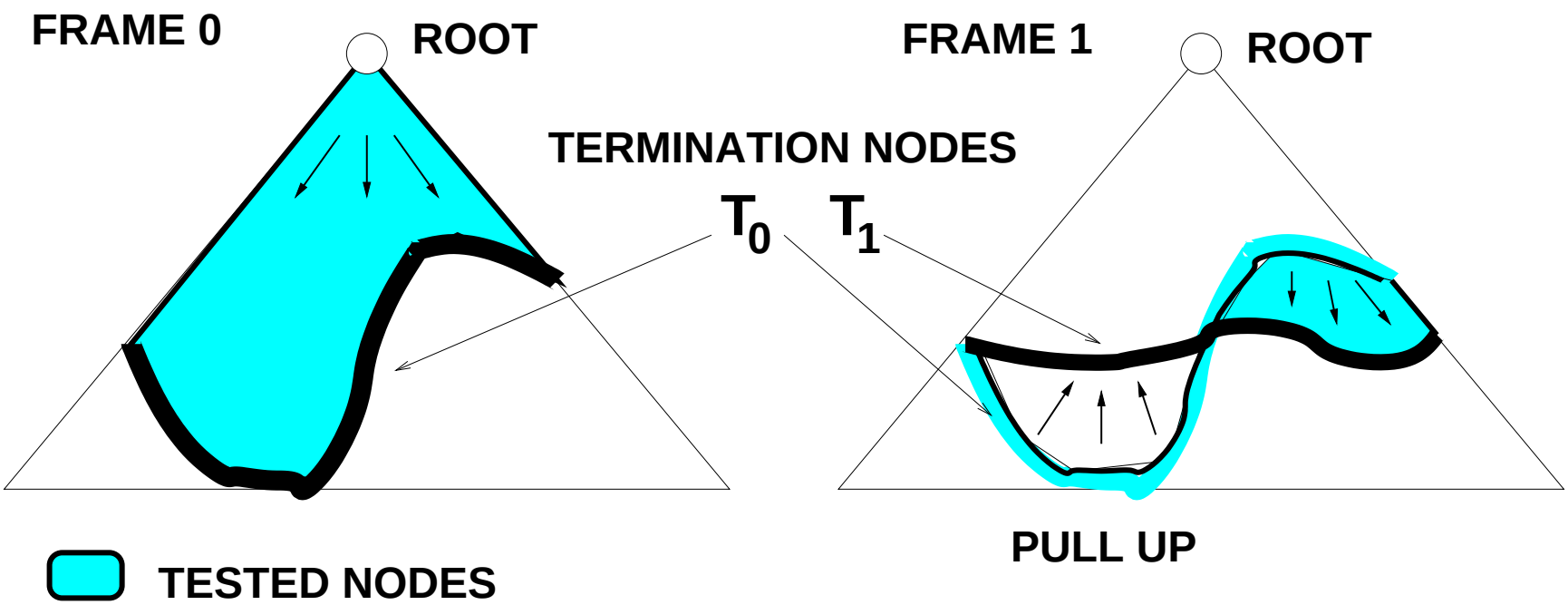
Object space

- Occlusion frusta. Zhang et al. ’97.
- Visual events. Coorg and Teller ’96, ’97.
- **Occlusion trees. Bittner et al. ’98.**

Hierarchy Updating

Computer Graphics Group

- idea: avoid repeated visibility tests of **interior** hierarchy nodes
- **cut of the hierachy** used as a starting level in next frame
- **propagate** visibility changes up the hierarchy



Conservative Hierarchy Updating

Computer Graphics Group

Motivation

- hierarchy updating saves almost 50% of visibility tests
- we can save more:
  partially visible and visible nodes **marked visible** with certain probability p

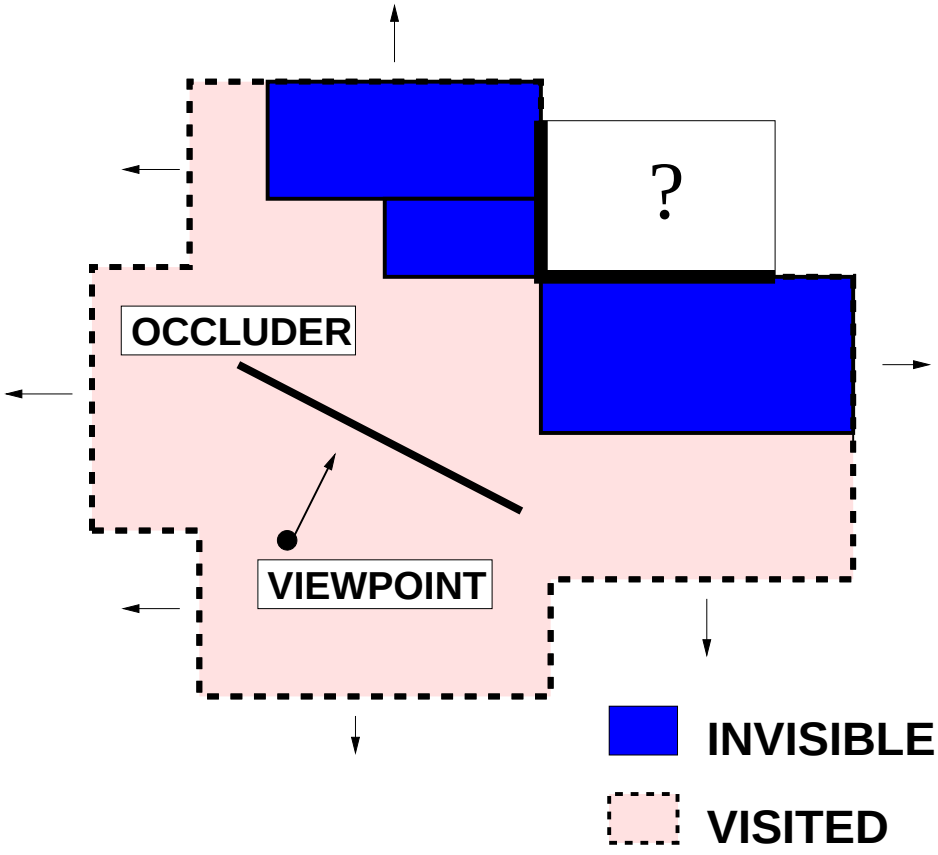
Properties

- keeps conservative behaviour of the algorithm
- not all changes from **PV** or **V** to **I** are captured
- higher p – **less** visibility tests, but usually **more** visible nodes and more objects to render

Visibility Propagation

Computer Graphics Group

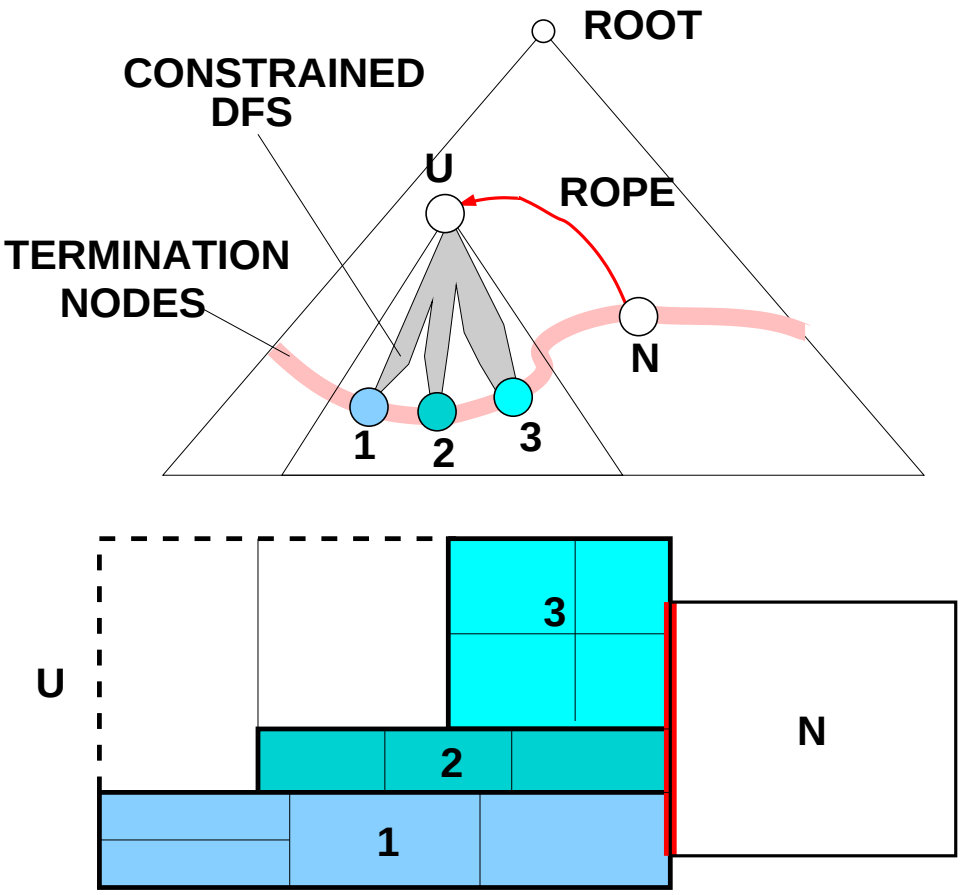
- **front–to–back** processing of the hierarchy nodes
- use visibility classification of already processed nodes
- combine their visibility – if **visible** or **invisible** we are done



Visibility Propagation – cont.

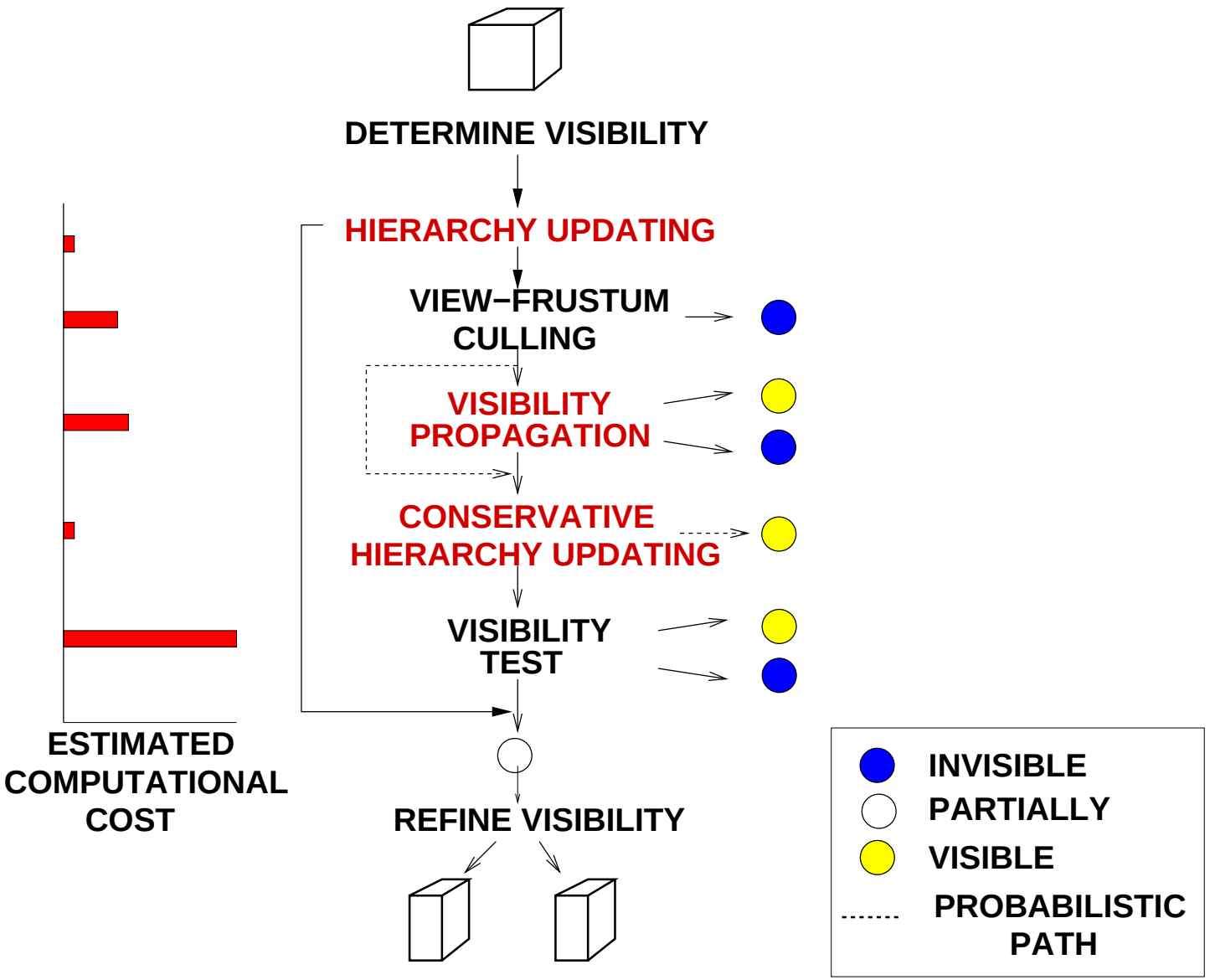
Computer Graphics Group

- use neighbour links (ropes) to locate all processed neighbours



Modified Visibility Pipeline

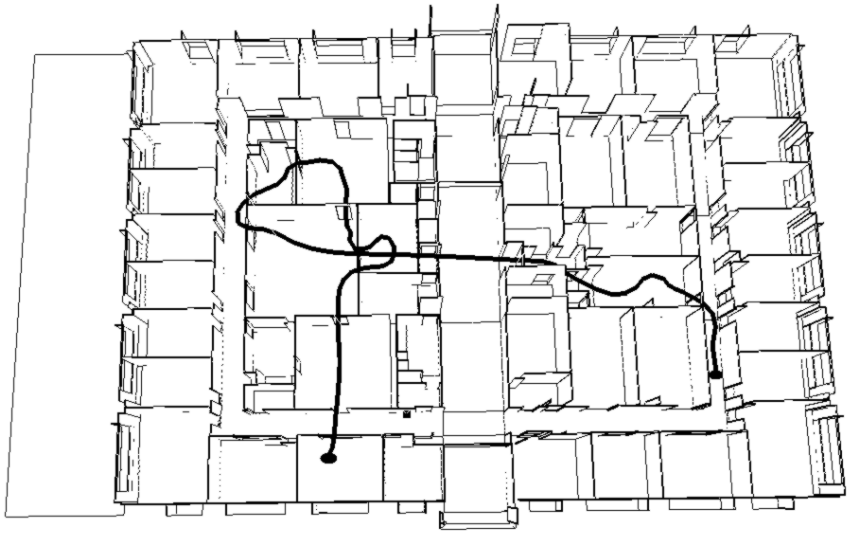
Computer Graphics Group



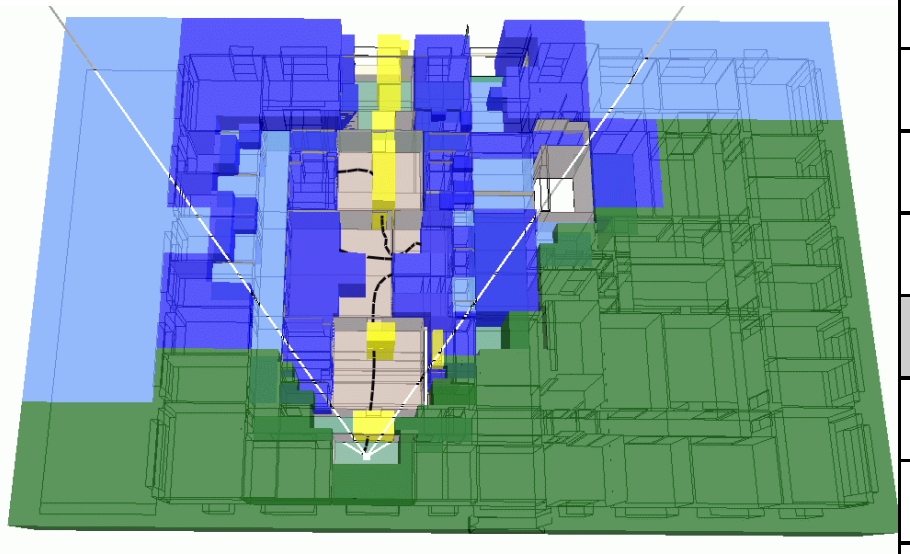
Results

Computer Graphics Group

Test path through building interior



Visibility classification



Results – cont.

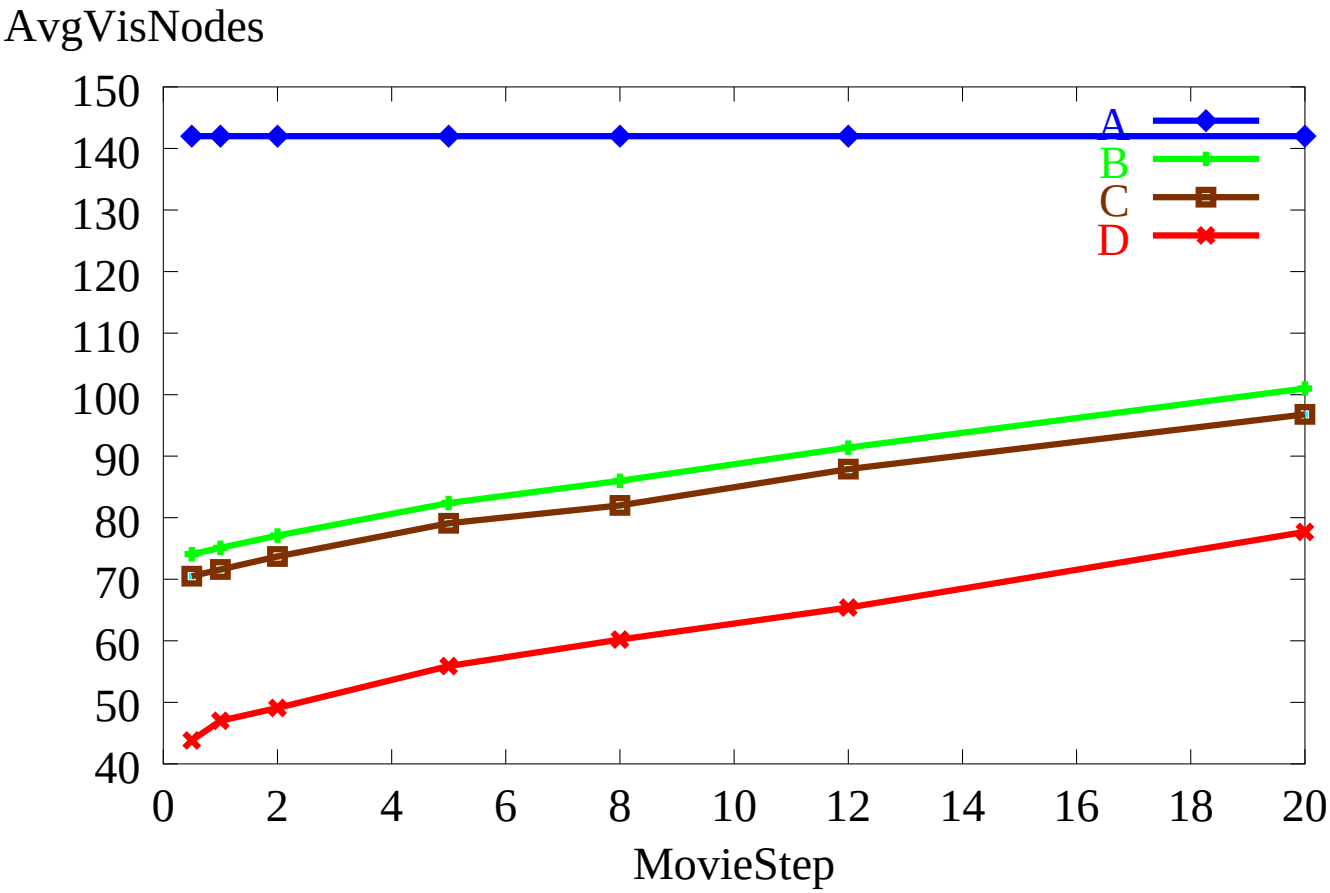
Computer Graphics Group

Walk through a building interior

Results – cont.

Computer Graphics Group

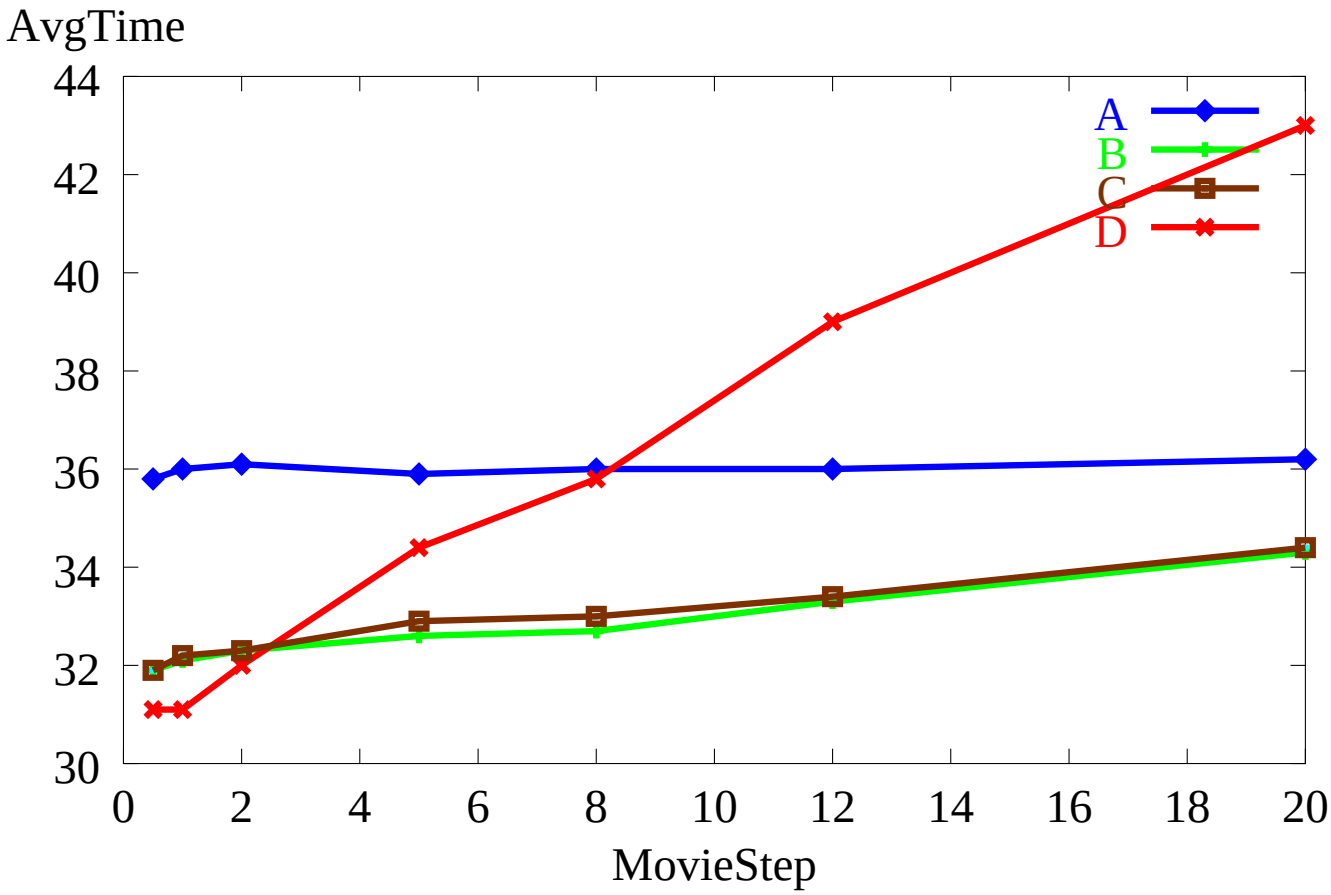
Avg. number of visibility tests vs. relative walking speed



Results – cont.

Computer Graphics Group

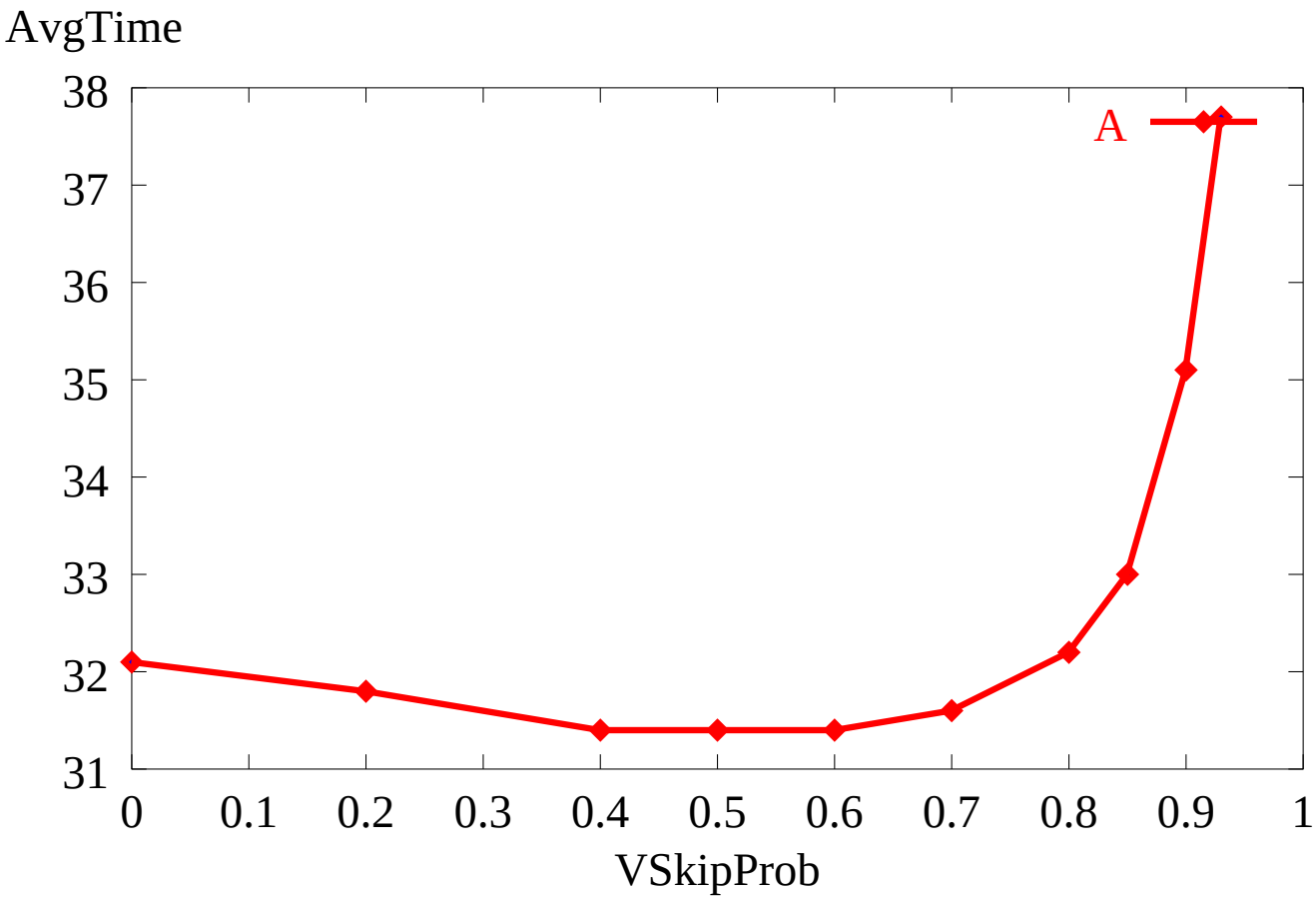
Avg. frame time [ms] vs. relative walking speed



Results – cont.

Computer Graphics Group

Avg. frame time [ms] vs. probability p



Results – summary

Computer Graphics Group

Hierarchy updating

- saves almost **50**% of visibility tests

Conservative hierarchy updating

- another **20**% of visibility tests
- faster visibility changes $\Rightarrow$ more objects to render

Visibility propagation

- successful on only few rather large regions

Conclusion and Future Work

Computer Graphics Group

Hierarchy updating

- very simple and useful
- eliminates overhead induced by hierachy, keeps advantages

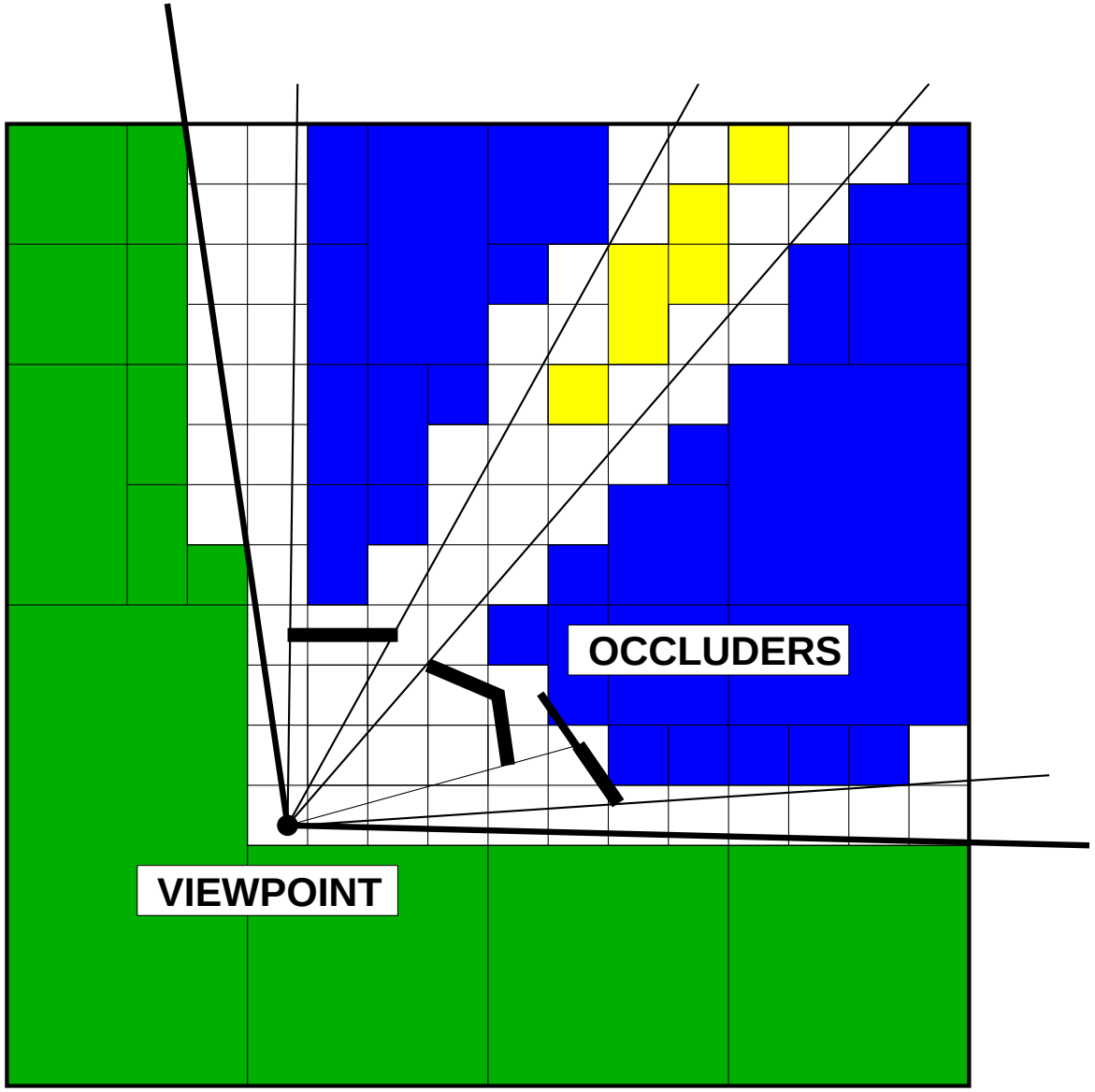
Conservative hierarchy updating

- useful for certain settings
- how to estimate **optimal probability** of skipping visibility test?

Visibility propagation

- not very succesful
- spatial coherence already exploited well by using the hierarchy!
- determine that region is **PV** with high probability?

QUESTIONS ?



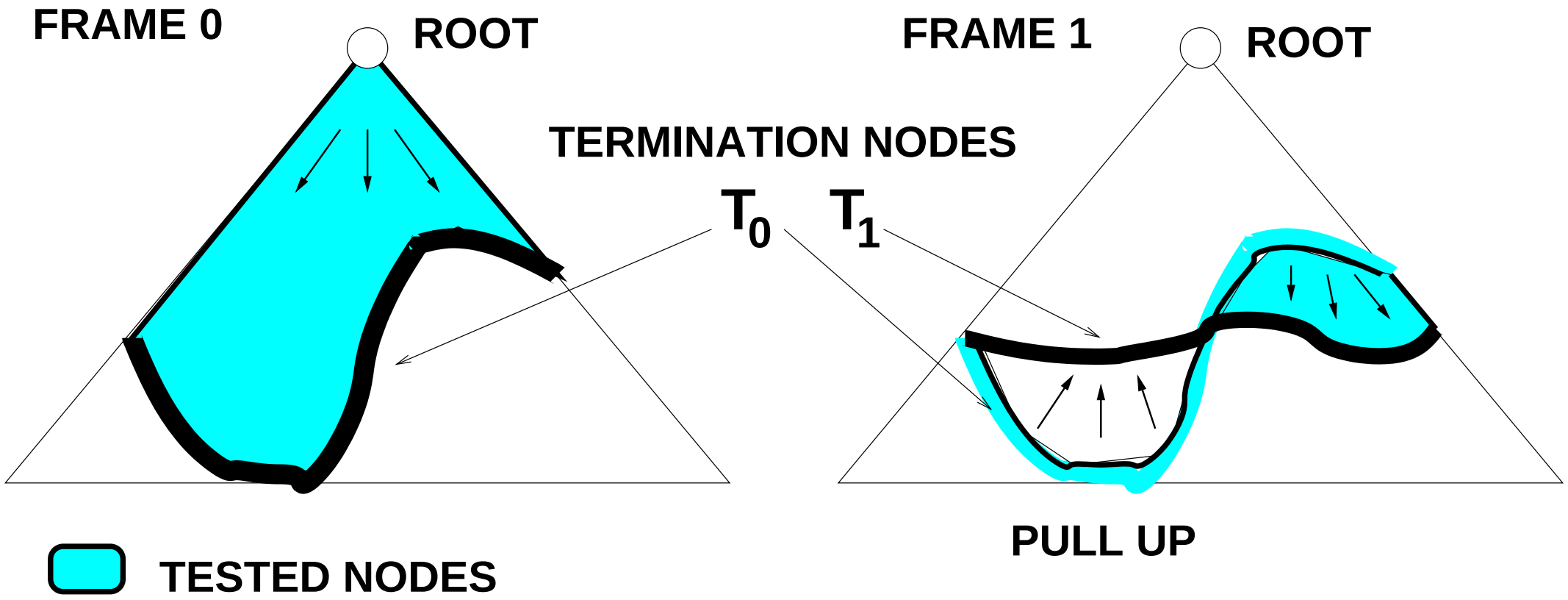
VISIBLE

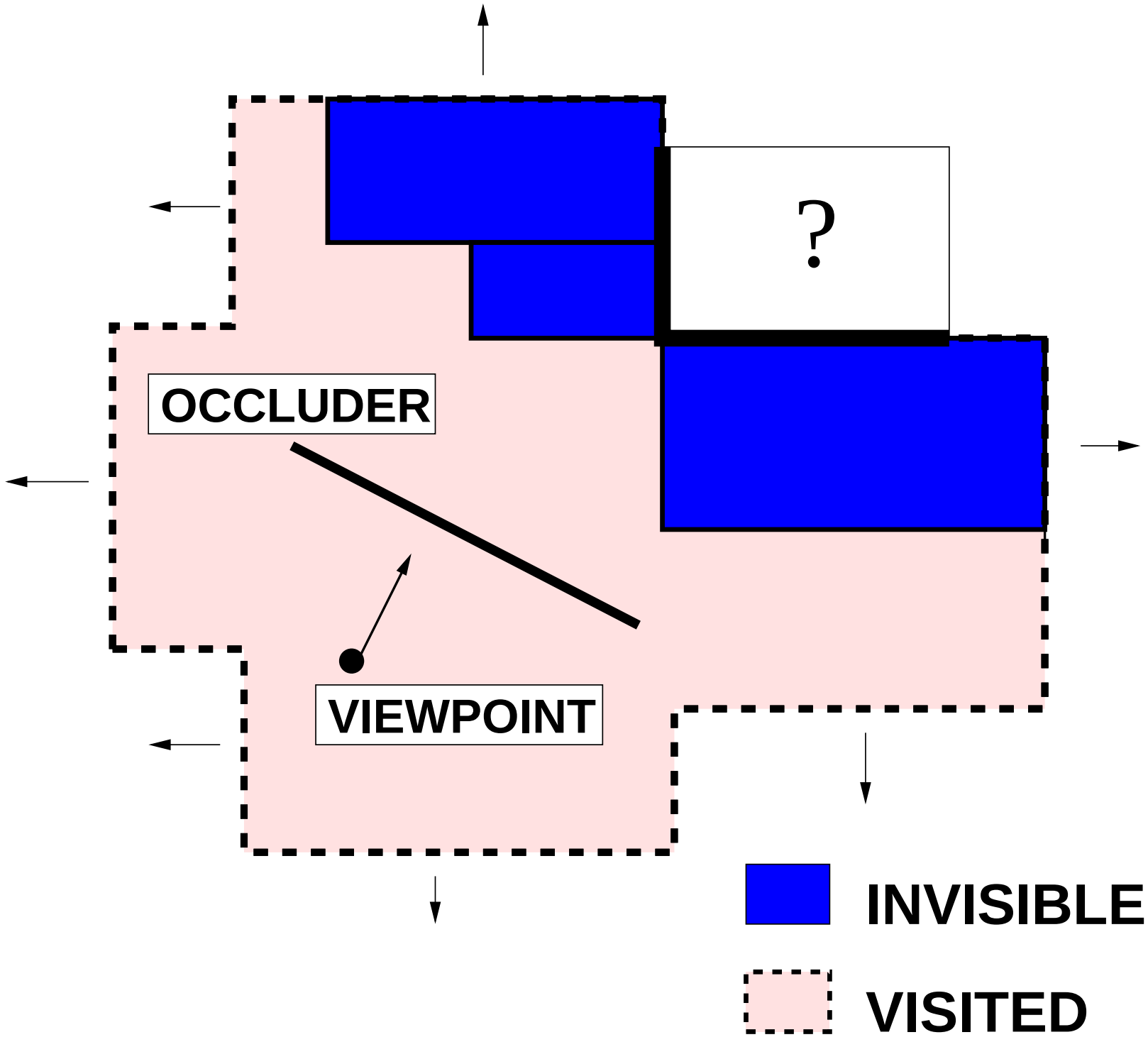
PARTIALLY

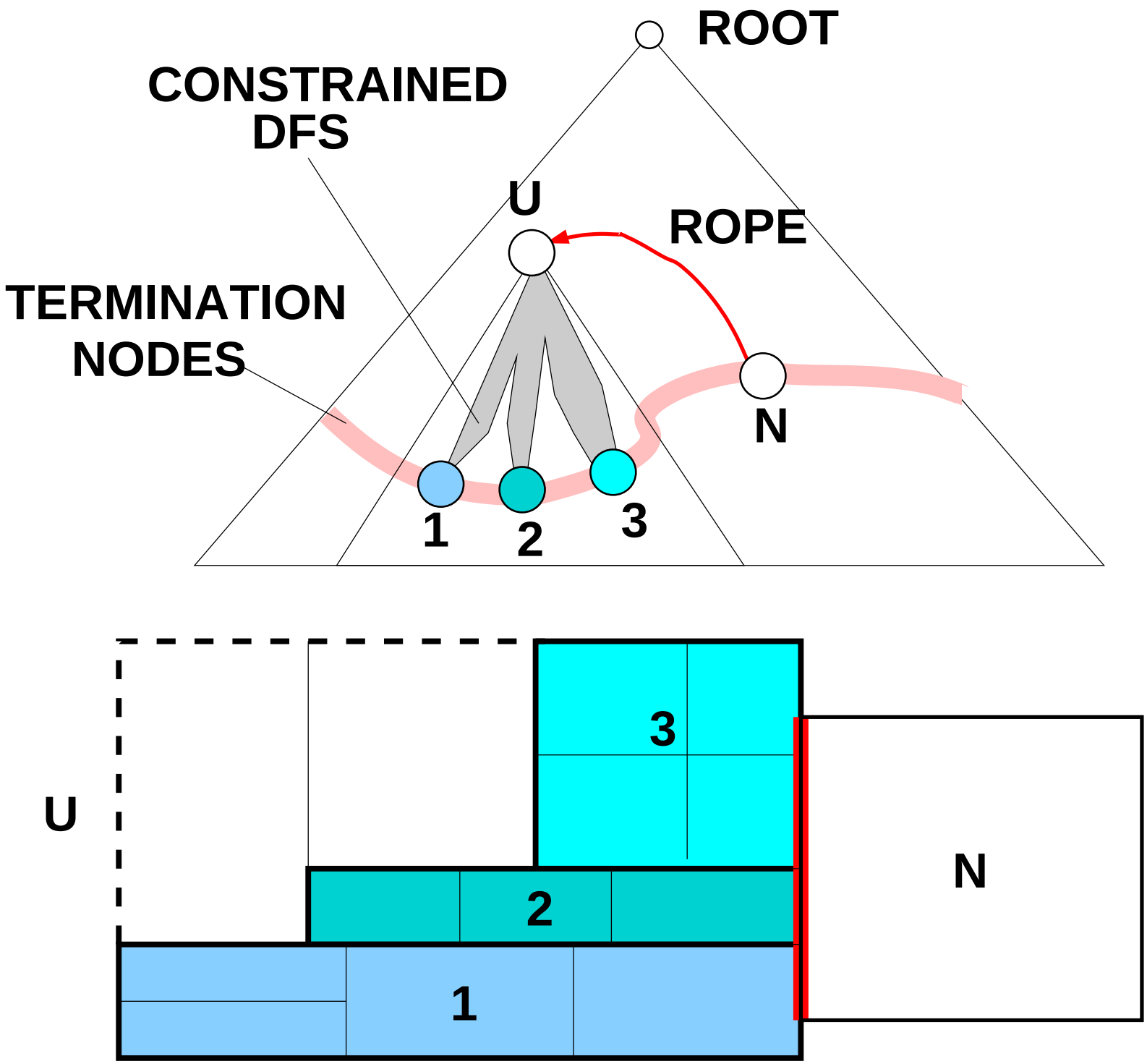
OCCLUDED

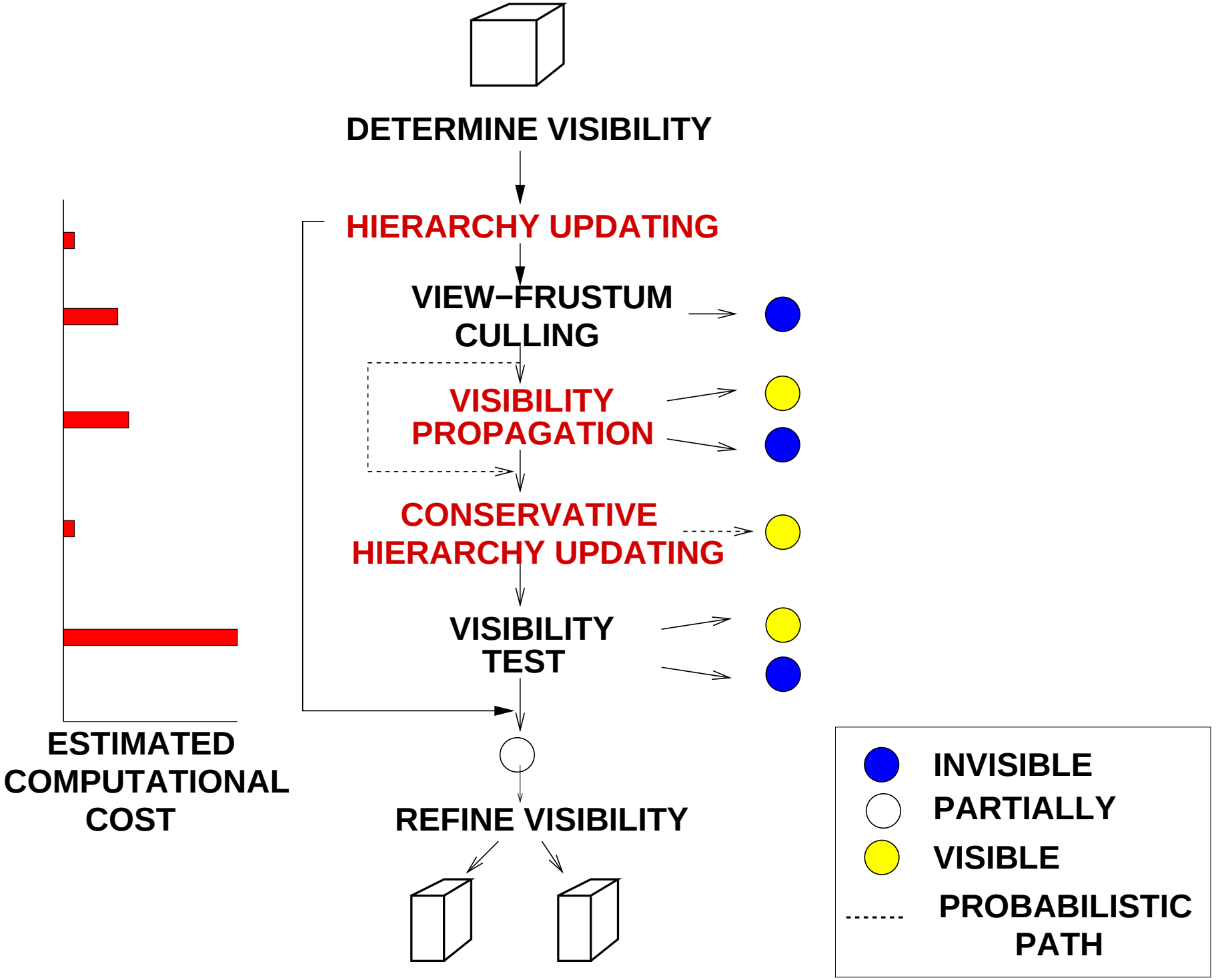
OUTSIDE VIEW−FRUSTUM

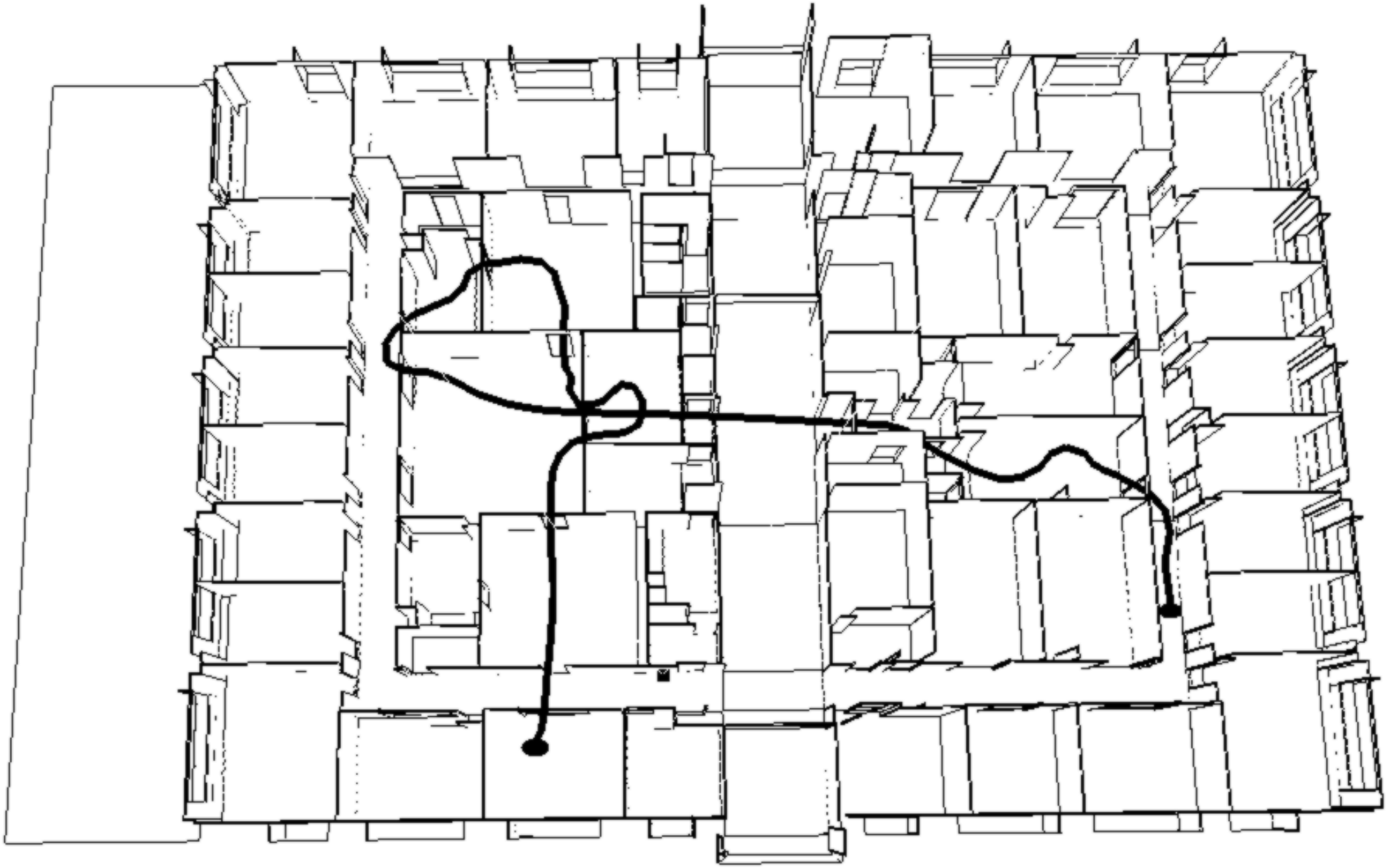
INVISIBLE

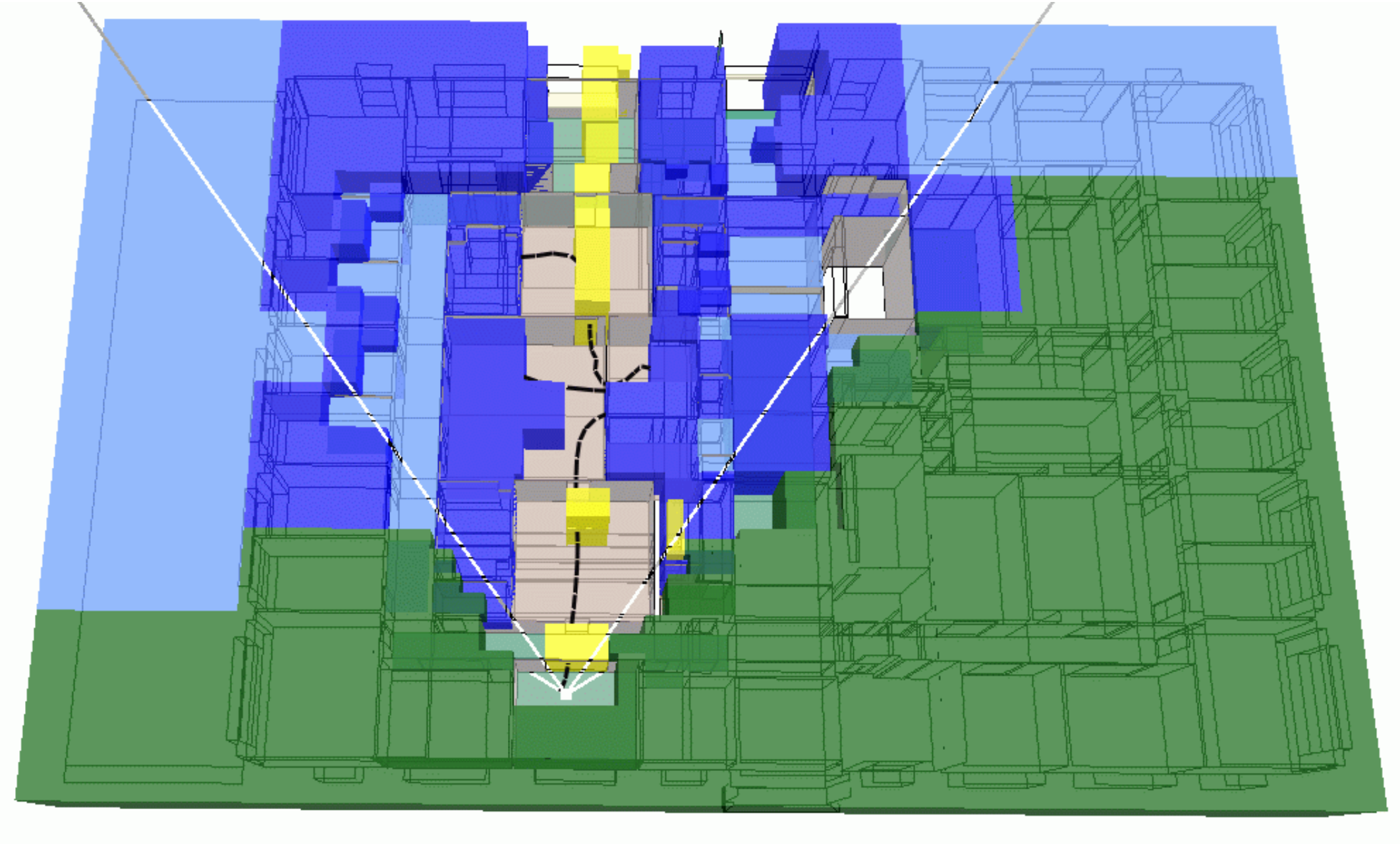

FRAME 0
ROOT
TERMINATION NODES
T_0
T_1
FRAME 1
ROOT
PULL UP
TESTED NODES

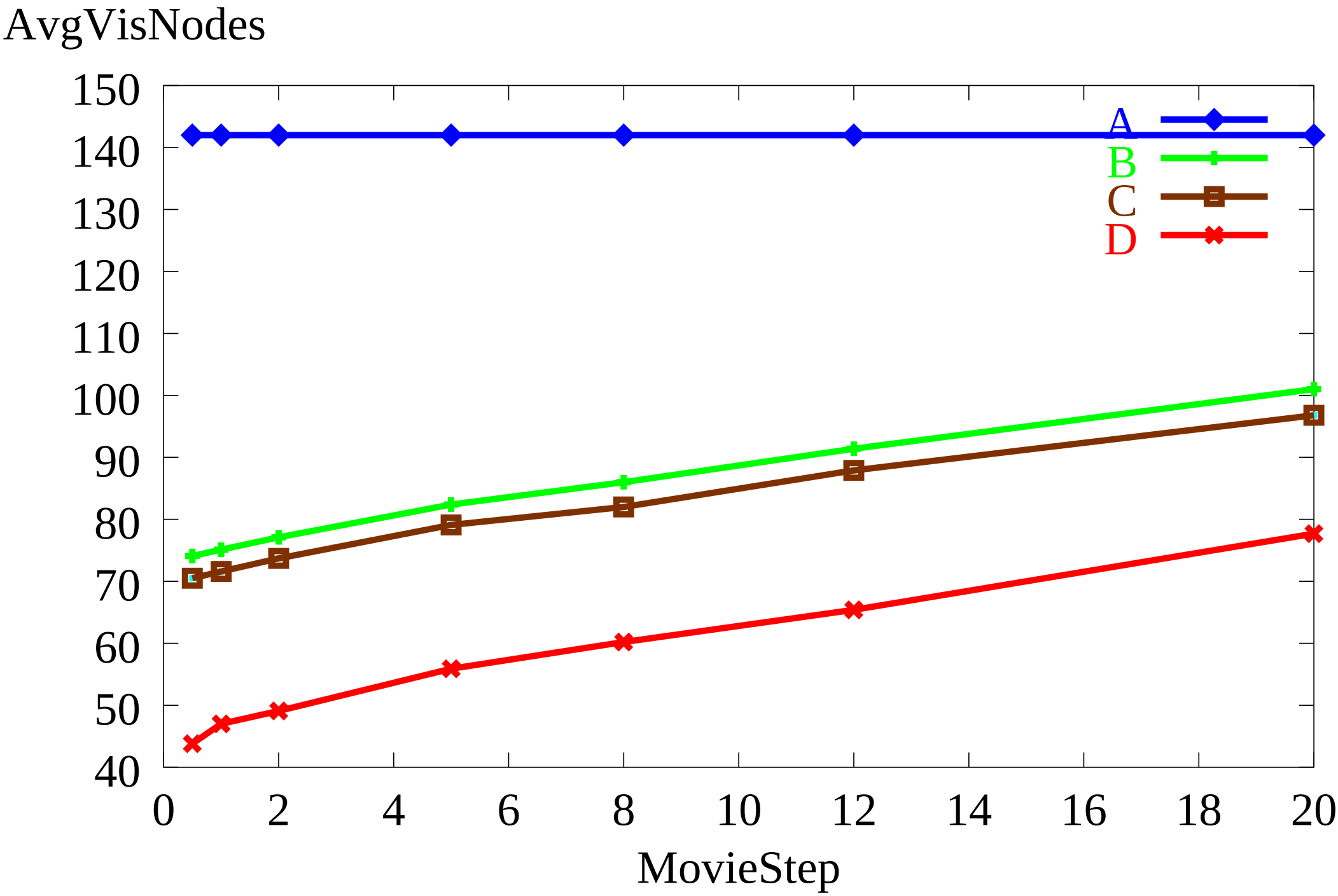

?
OCCLUDER
VIEWPOINT
INVISIBLE
VISITED

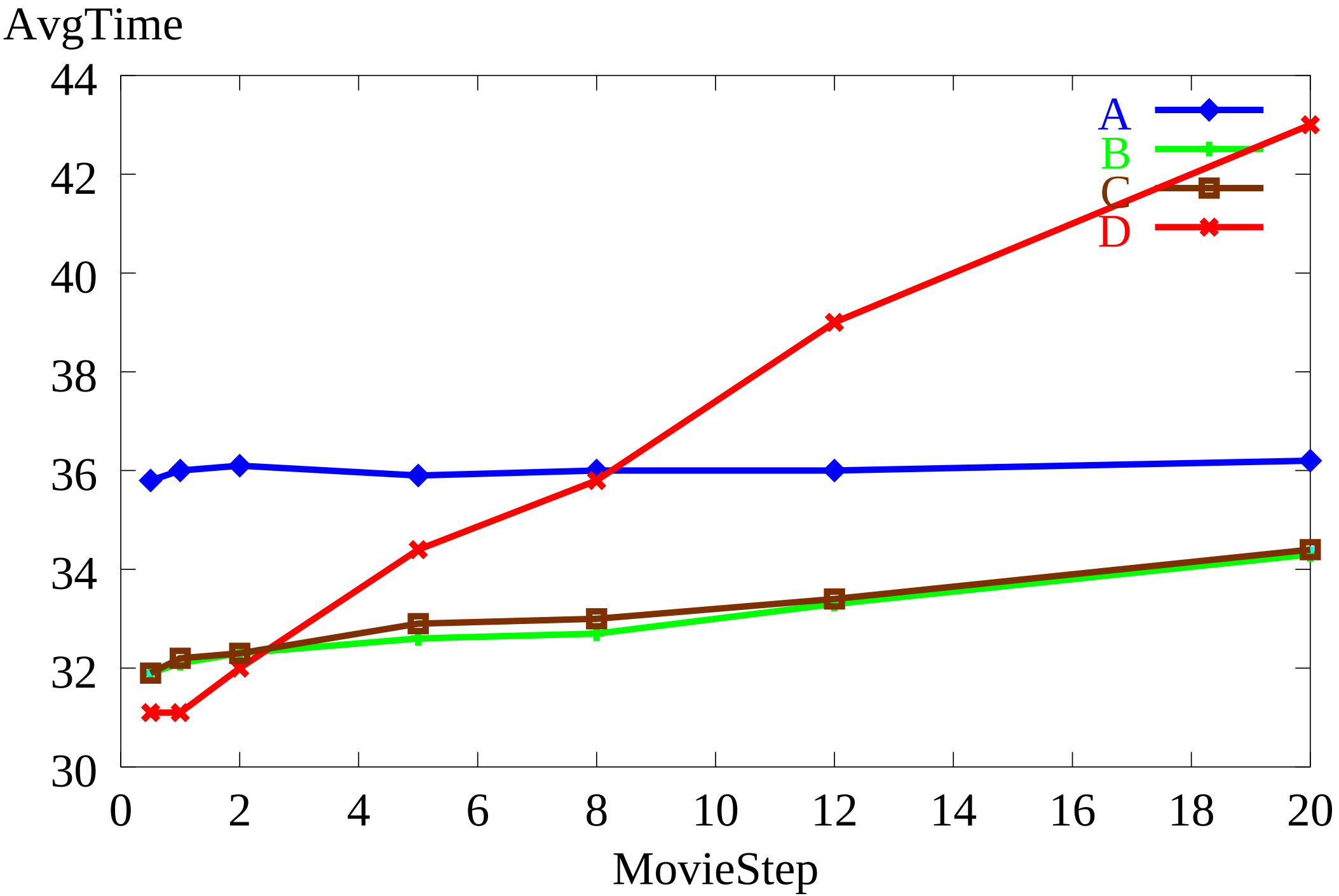

ROOT
CONSTRAINED
DFS
U
ROPE
TERMINATION
NODES
N
1
2
3
U
3
2
1
N

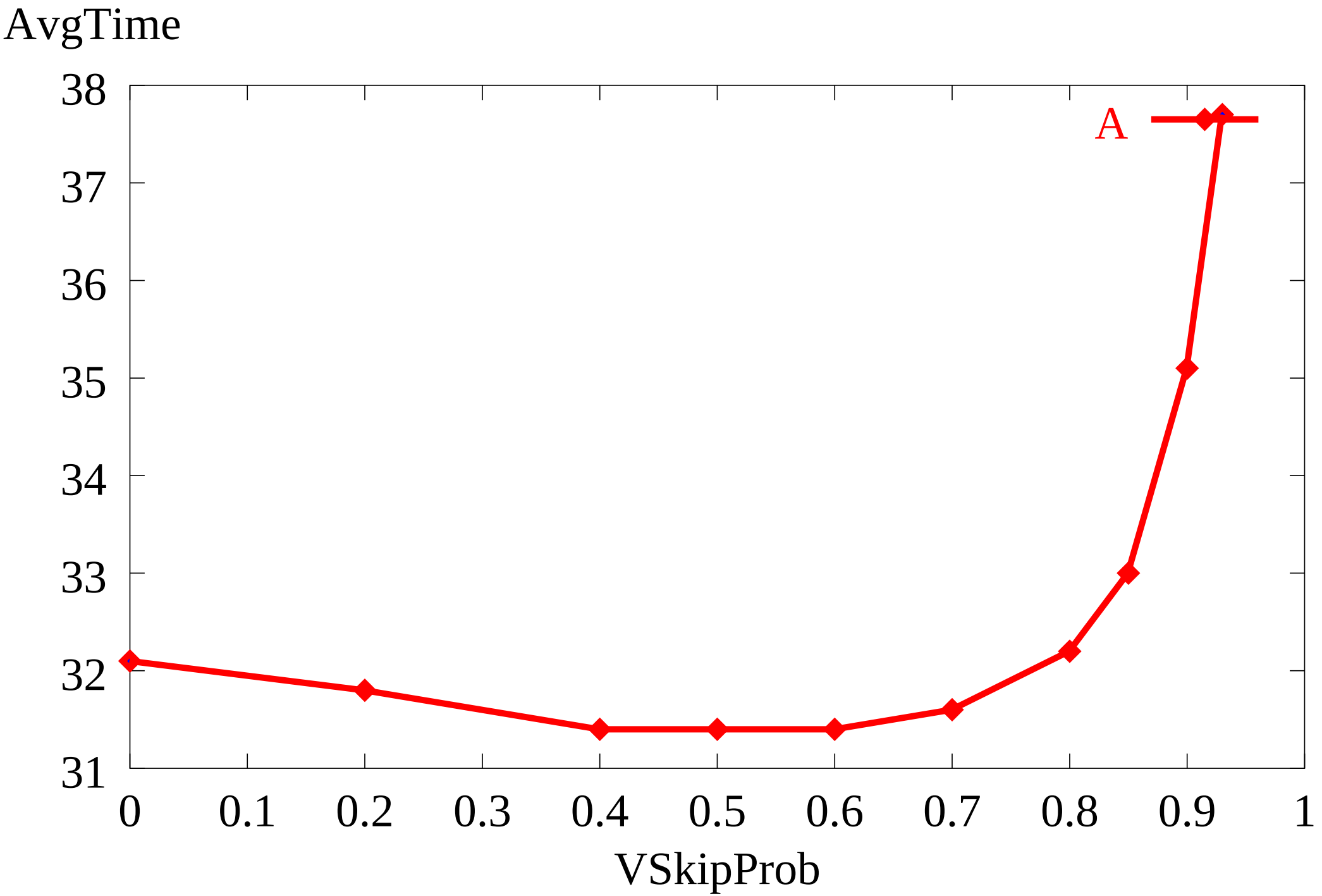

DETERMINE VISIBILITY
HIERARCHY UPDATING
VIEW−FRUSTUM CULLING
VISIBILITY PROPAGATION
CONSERVATIVE HIERARCHY UPDATING
VISIBILITY TEST
REFINE VISIBILITY
ESTIMATED COMPUTATIONAL COST
INVISIBLE
PARTIALLY
VISIBLE
PROBABILISTIC PATH






AvgVisNodes
150
140
130
120
110
100
90
80
70
60
50
40
0
2
4
6
8
10
12
14
16
18
20
MovieStep
A
B
C
D


AvgTime
44
42
40
38
36
34
32
30
0
2
4
6
8
10
12
14
16
18
20
MovieStep
A
B
C
D


AvgTime
38
37
36
35
34
33
32
31
A
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
VSkipProb

[back]