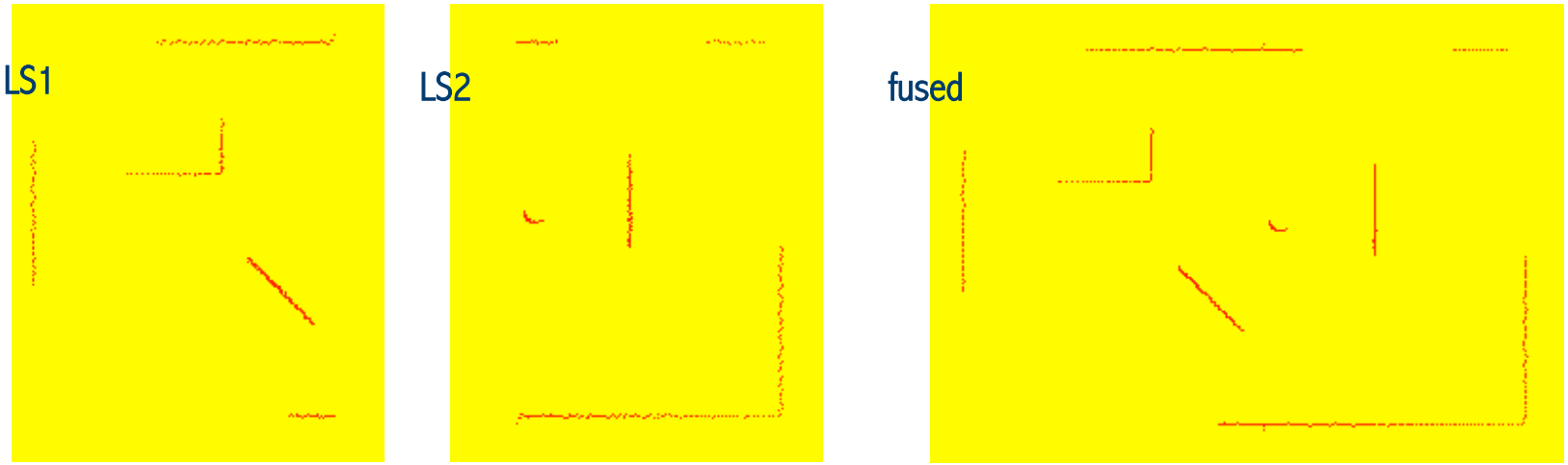
Application Example: Motion Control

Motion Control

- is one of the most important tasks of mobile embedded systems
- is subject to real-time and reliability requirements
- heavily depends on the sensory input

- Single sensors have a partial, inaccurate view
- Distributed sensor fusion allows achieving a more complete and accurate perception of the environment



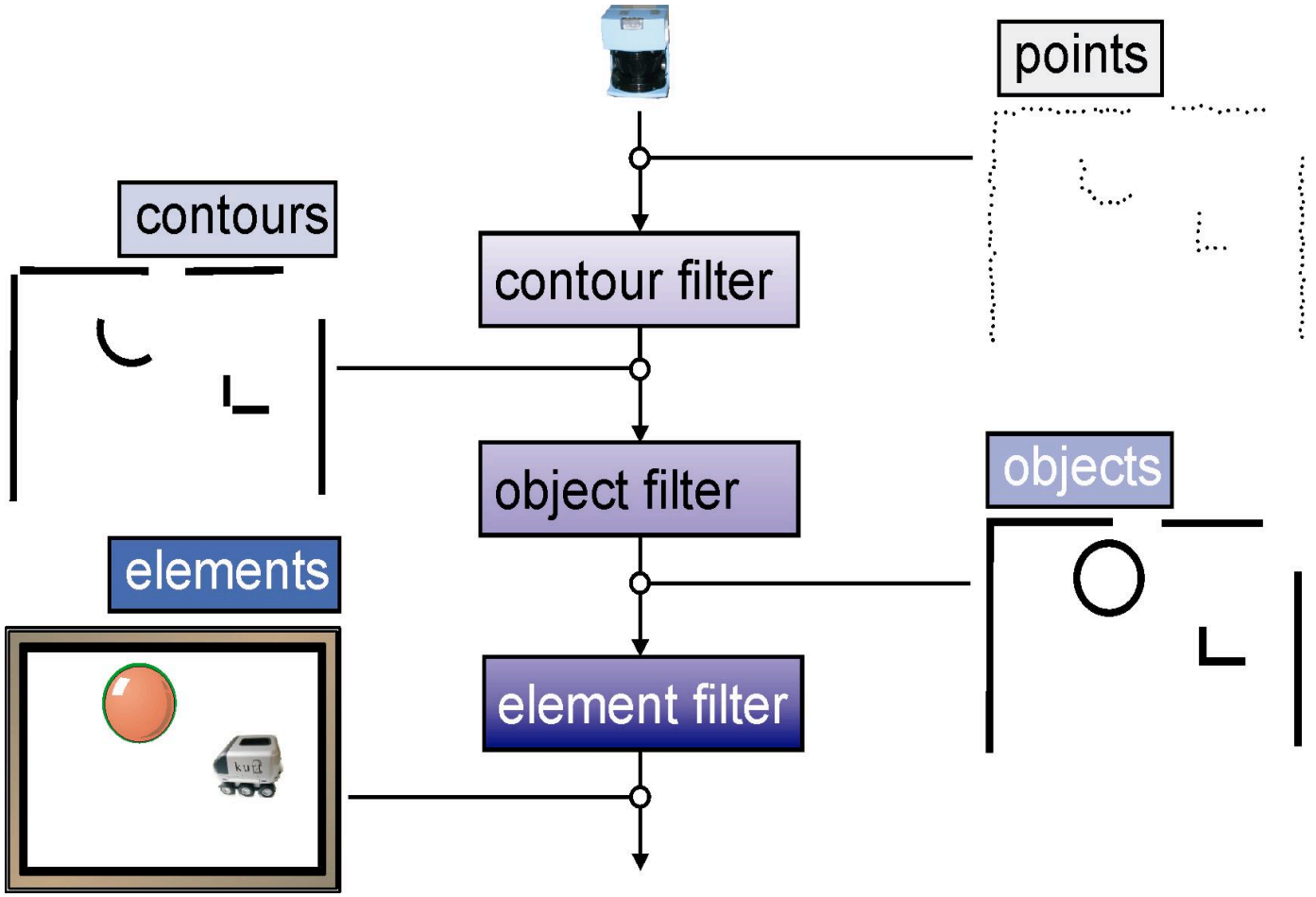
Example: RoboCup

- Getting wide acceptance as standard benchmark for team robotic
- Annual world and national championships in robot soccer
- Research done as part of a nationwide DFG-program „Cooperating teams of mobile robots in dynamic environments“

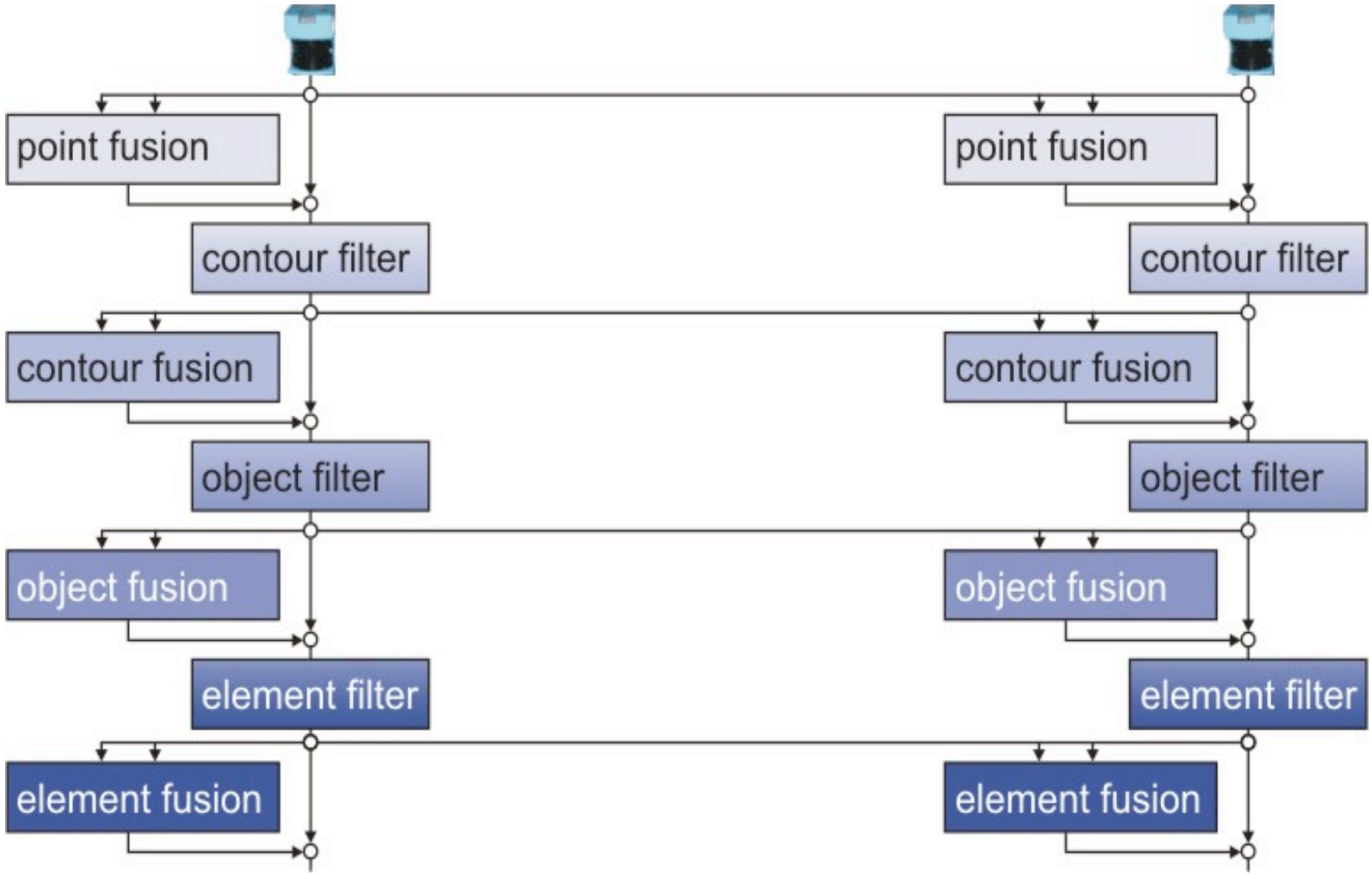
Example:RoboCup

Where is the ball?

Sensor Data Processing – Filters and Abstraction Levels

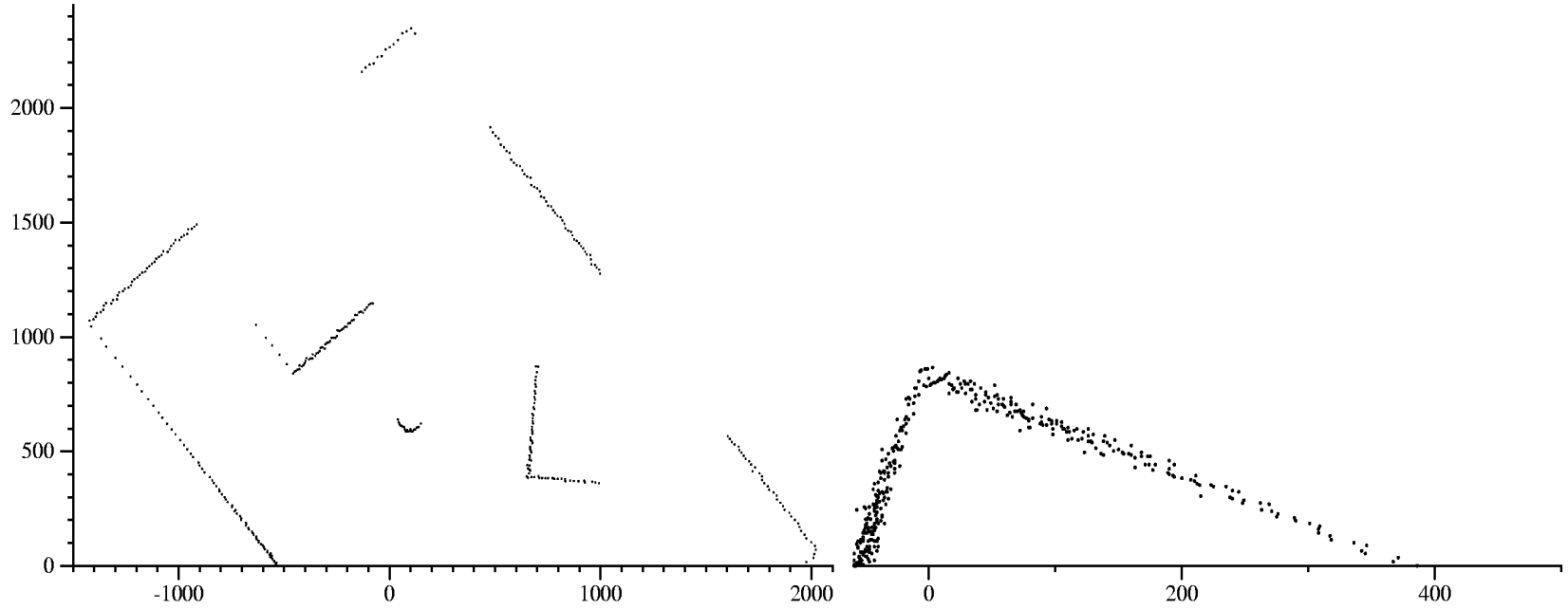


Sensor Data Processing - Multi-Level Sensor Fusion

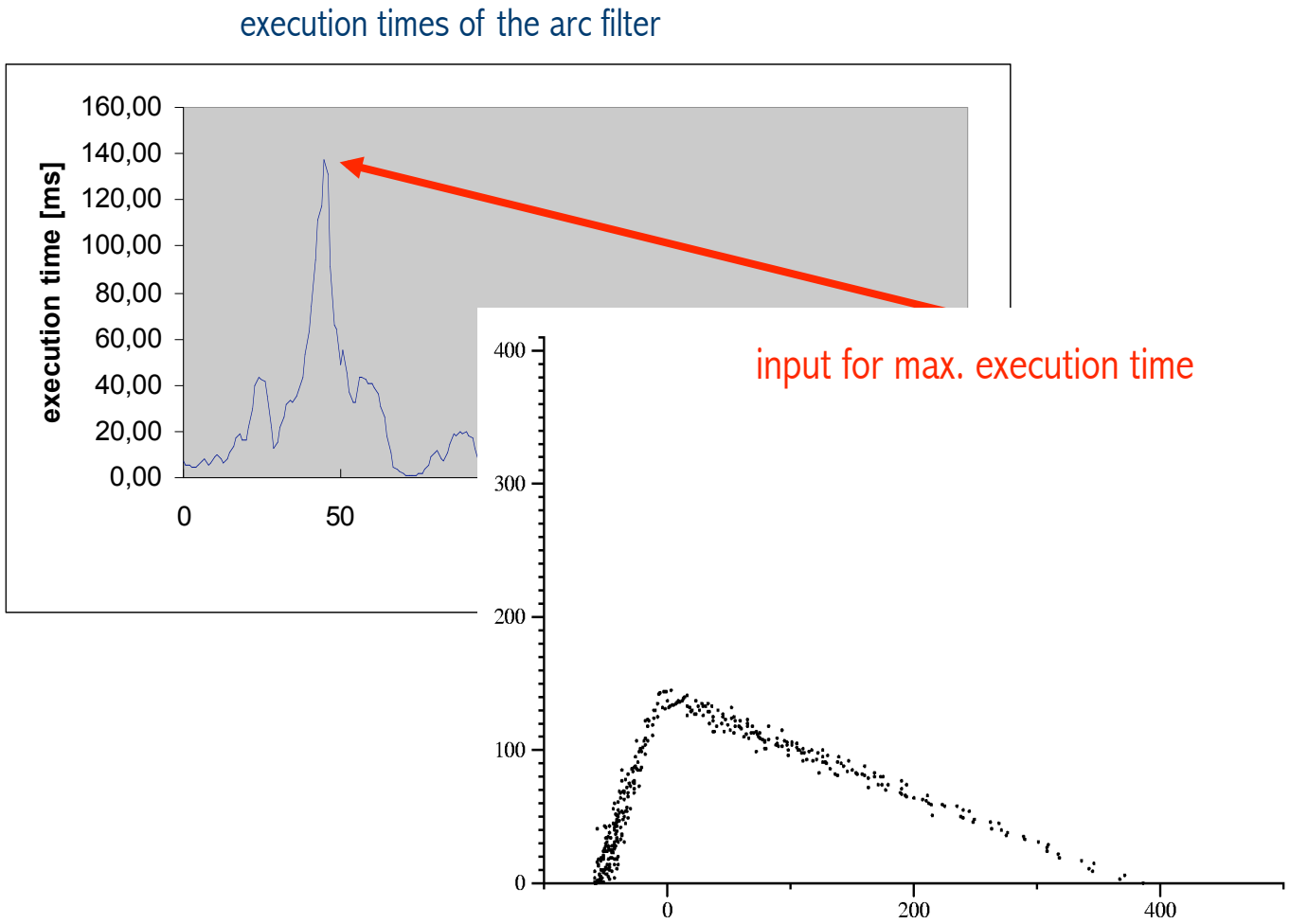


Environment-Dependent Execution Times of the Filter Modules

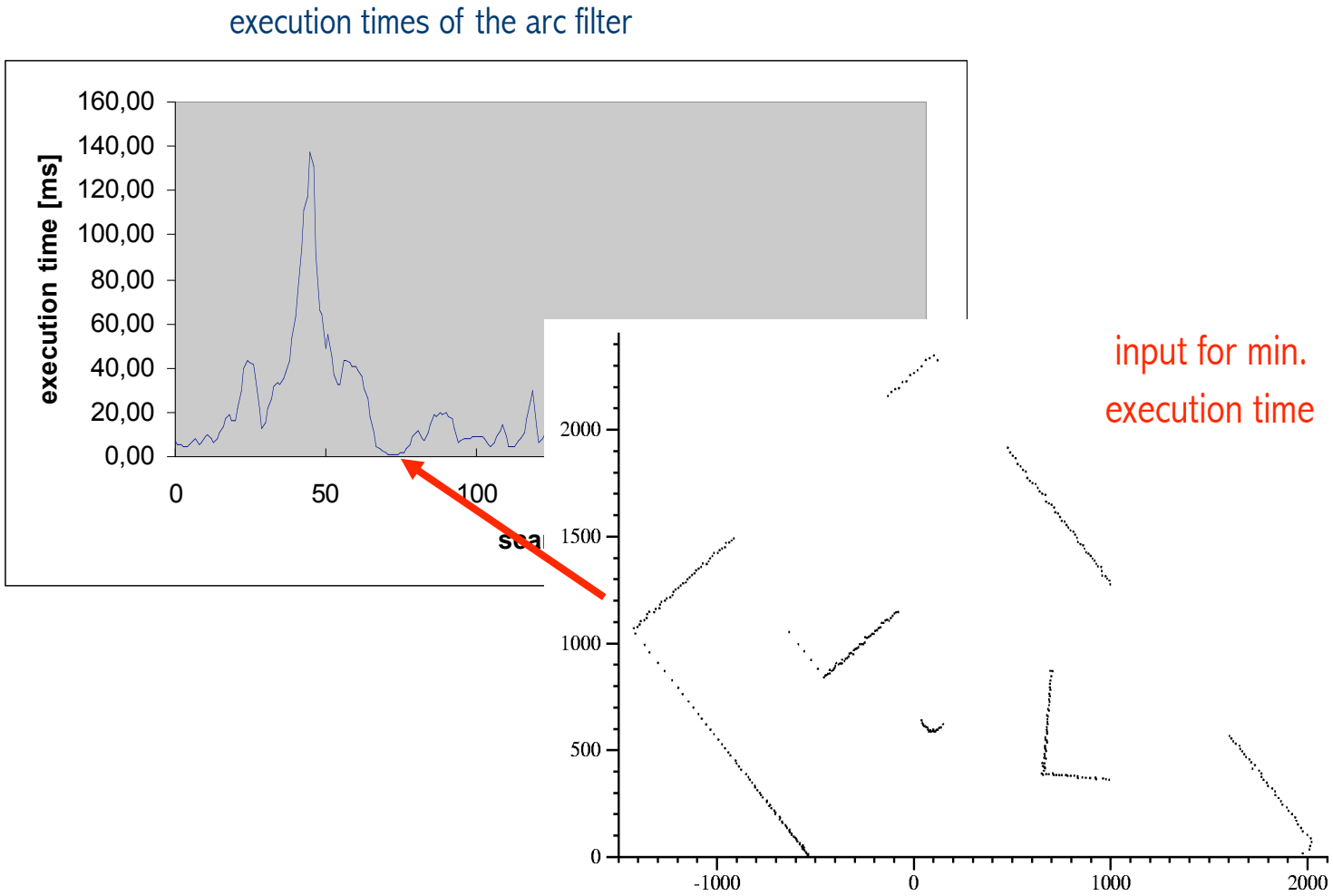
- Execution times depend on input size
- Execution times depend on input content



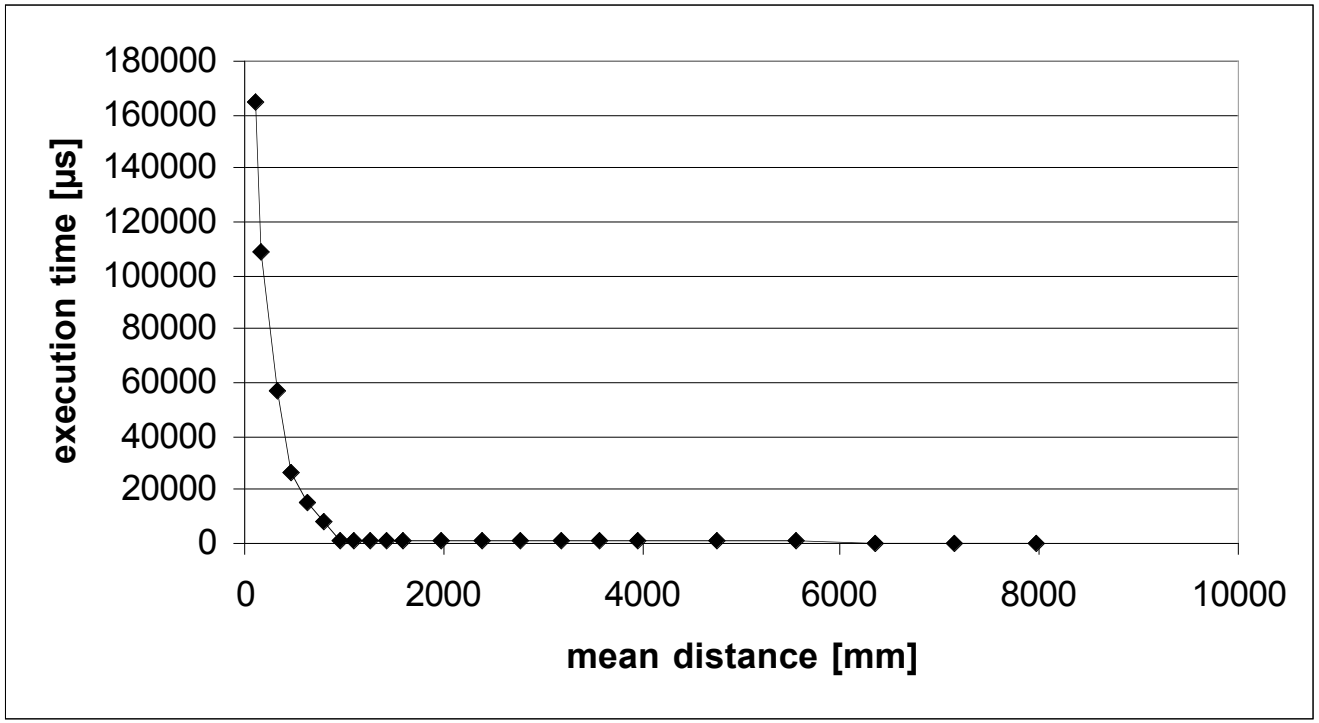
Environment-Dependent Execution Times of the Filter Modules (cont'd)



Environment-Dependent Execution Times of the Filter Modules (cont'd)



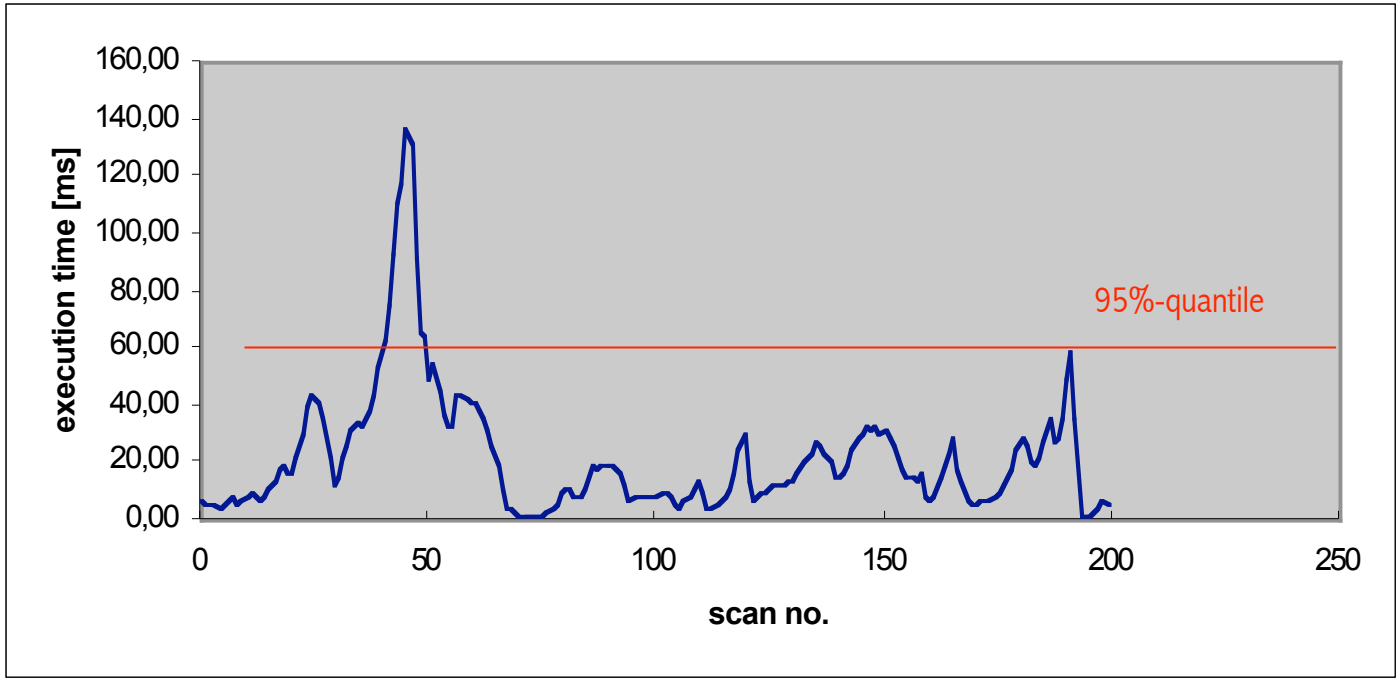
Contour Filter: Execution Time vs. Mean Distance



- Execution times > 57ms are only observed for distances < 32cm

Environment-Dependent Execution Times of the Filter Modules (cont'd)

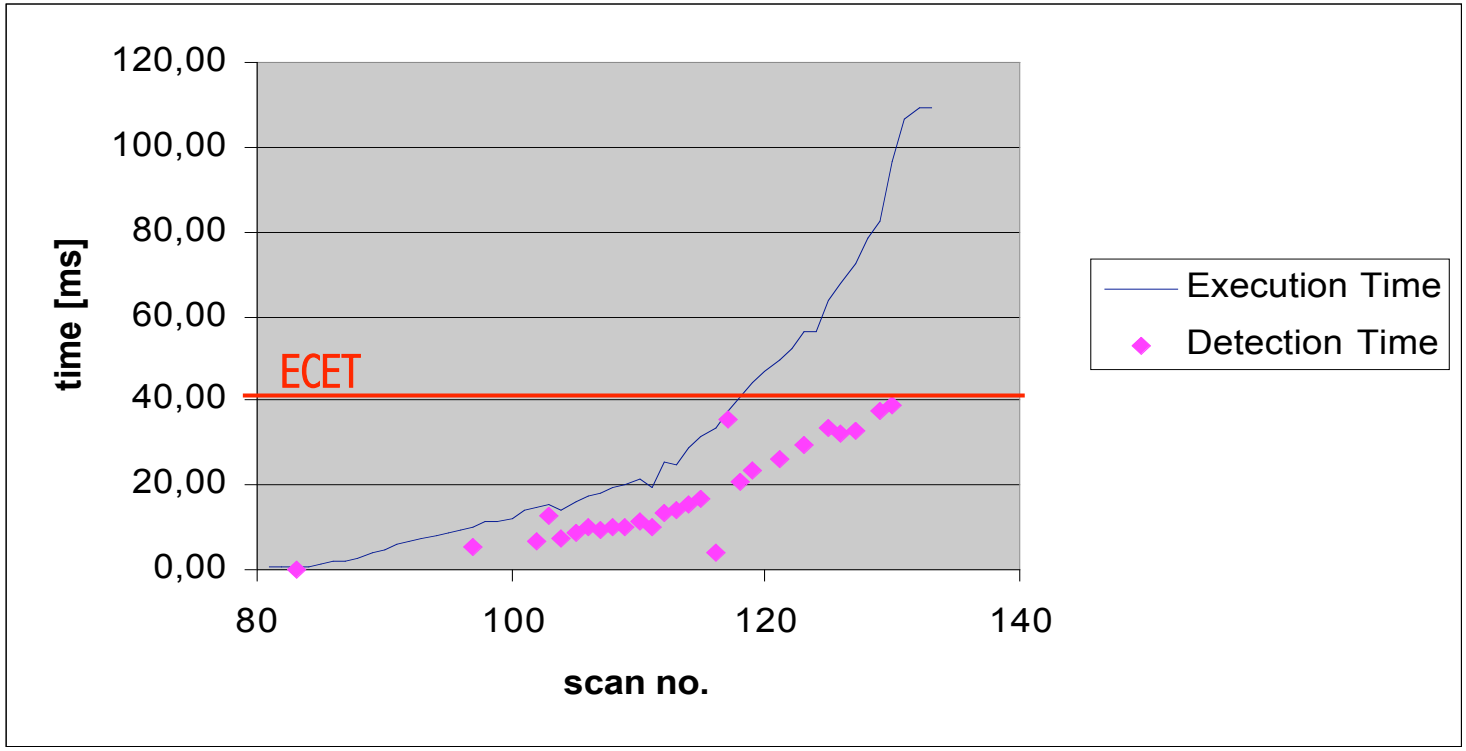
execution times of the arc filter



→ Scheduling WCETs is not an acceptable solution to achieve a predictable timing behavior of the filter modules!

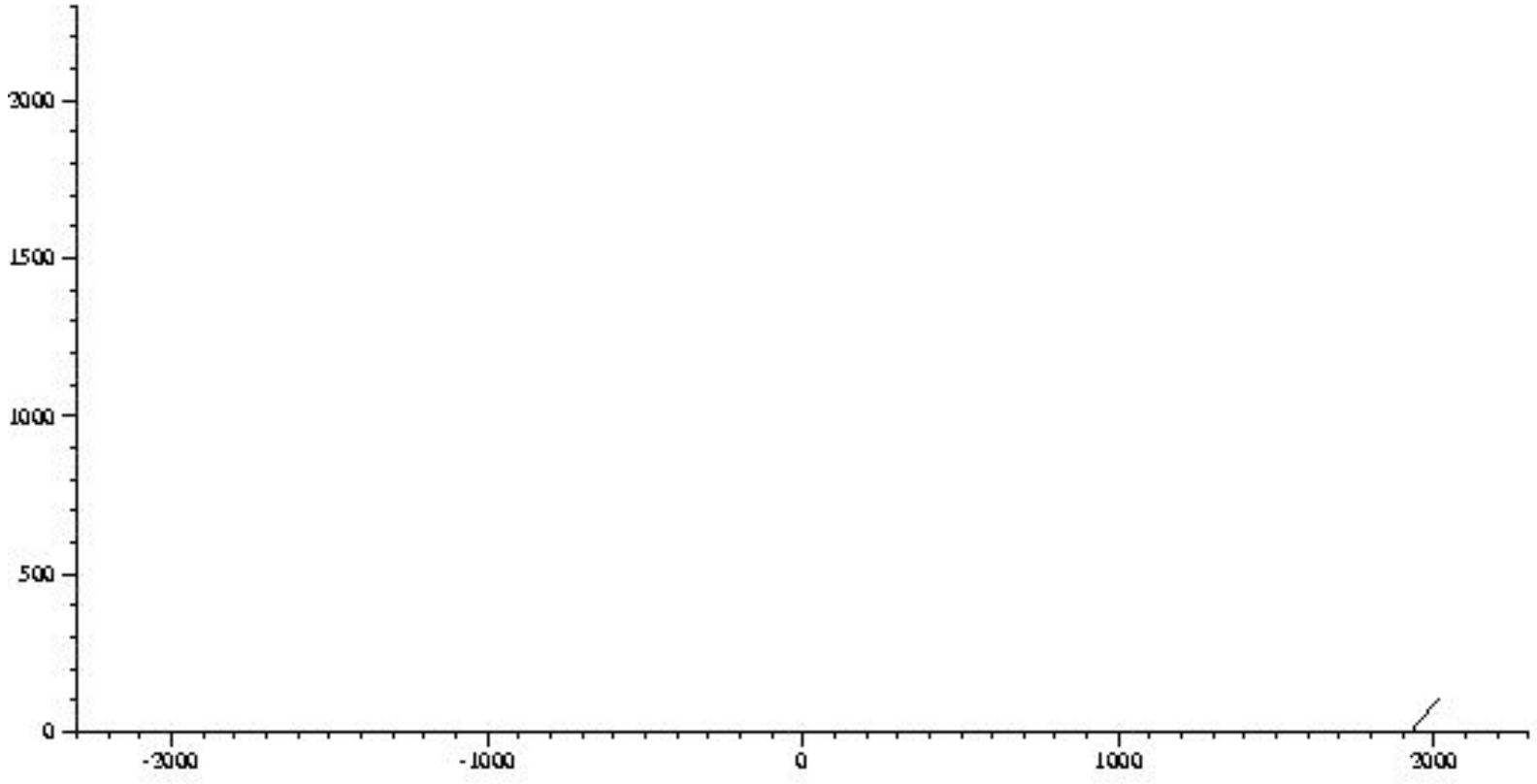
Functional Redundancy in the Arc Filter

- The arc filter evaluates potential ball positions
- Only the best estimate is relevant to higher layers (corresponds to ball)



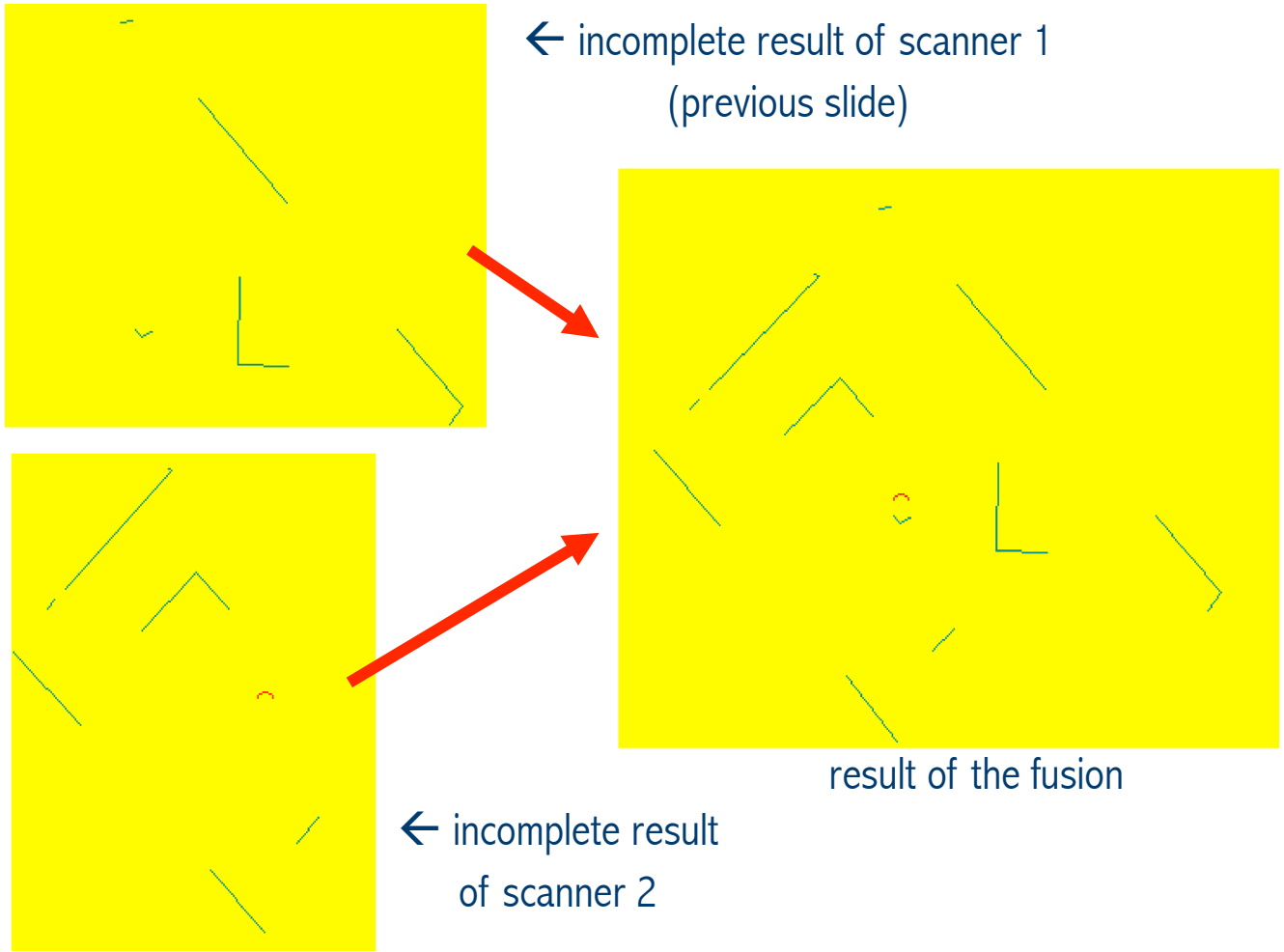
Structural Redundancy in the Edge Filter

- Aborted instances deliver results for a fraction of the scene
- Incomplete results are valuable input for the sensor fusion



Structural Redundancy – Distributed Fusion

- Environment is observed by several, distributed sensors
- Fusion complements missing results from one sensor by results from others

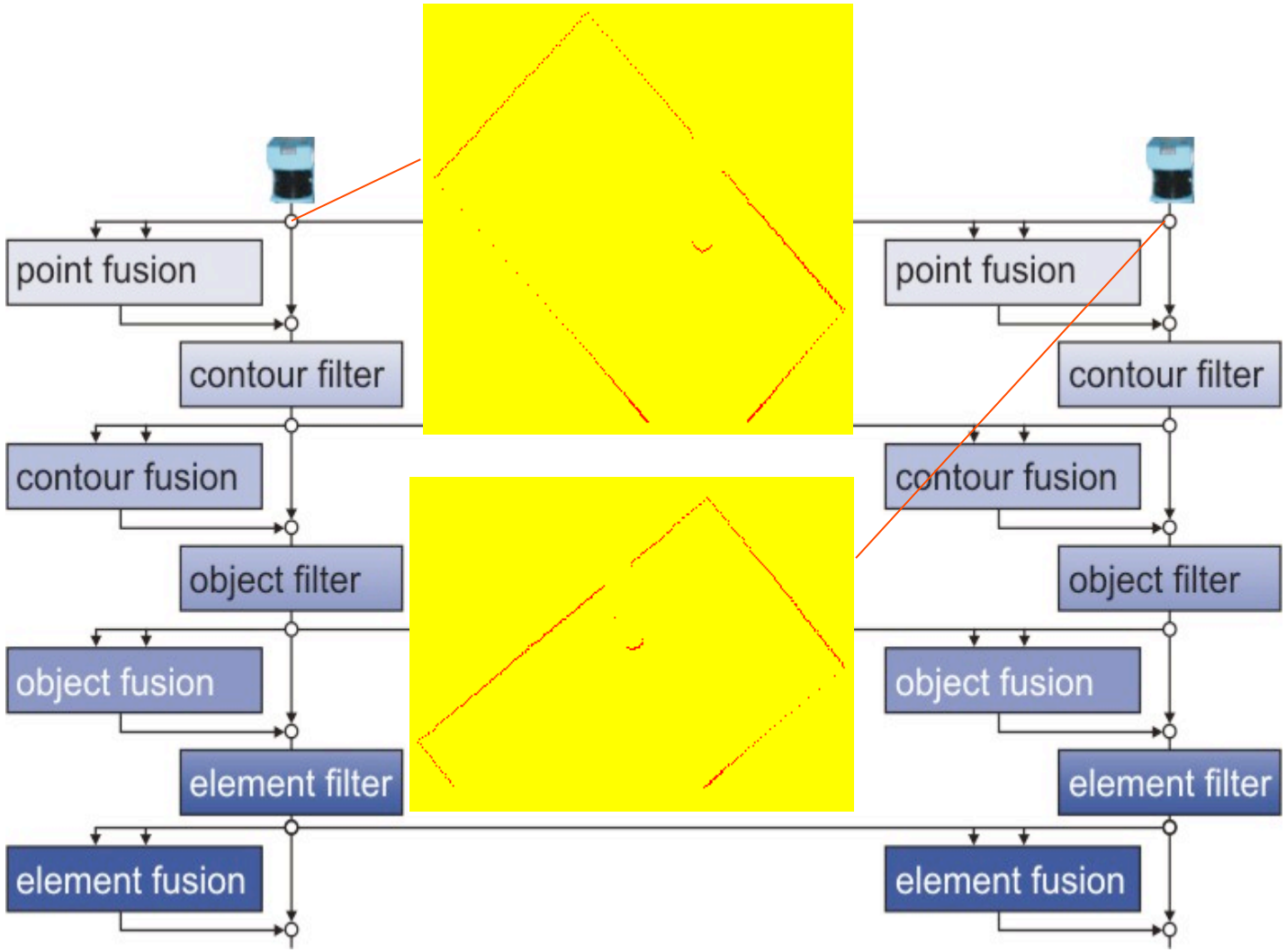


Application-Level Adaptation

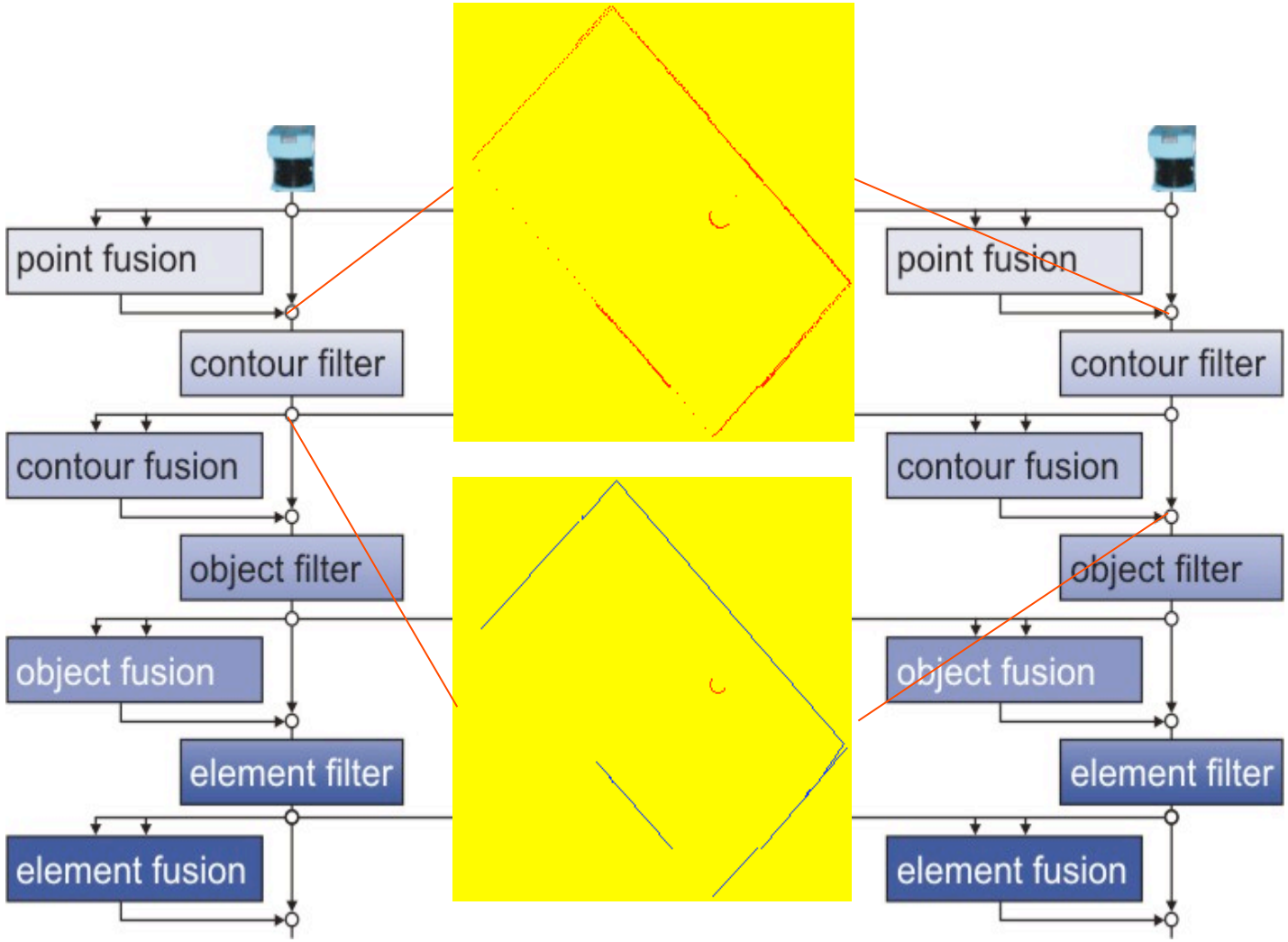
- Raising the fusion level allows reducing system load
  - Reduced execution times (CPU load)
  - Reduced data volume (network load)
- Raising the fusion level may decrease accurateness of results

		2 scanners		3 sanners	
		execution times [µs]	data volume [byte]	execution times [µs]	data volume [byte]
point fusion	point fusion	10304		20427	
	arc filter	13157		63093	
	edge filter	1436		2216	
	obect filter	2570		3708	
	element filter	77		98	
	sum	27544	11616	89542	17424
contour fusion	arc filter	3398 3688		3398 3688 5428	
	edge filter	577 578		577 578 465	
	contour fusion	1955		3253	
	object filter	1636		2067	
	element filter	70		77	
	sum	7927	1384	11290	1912

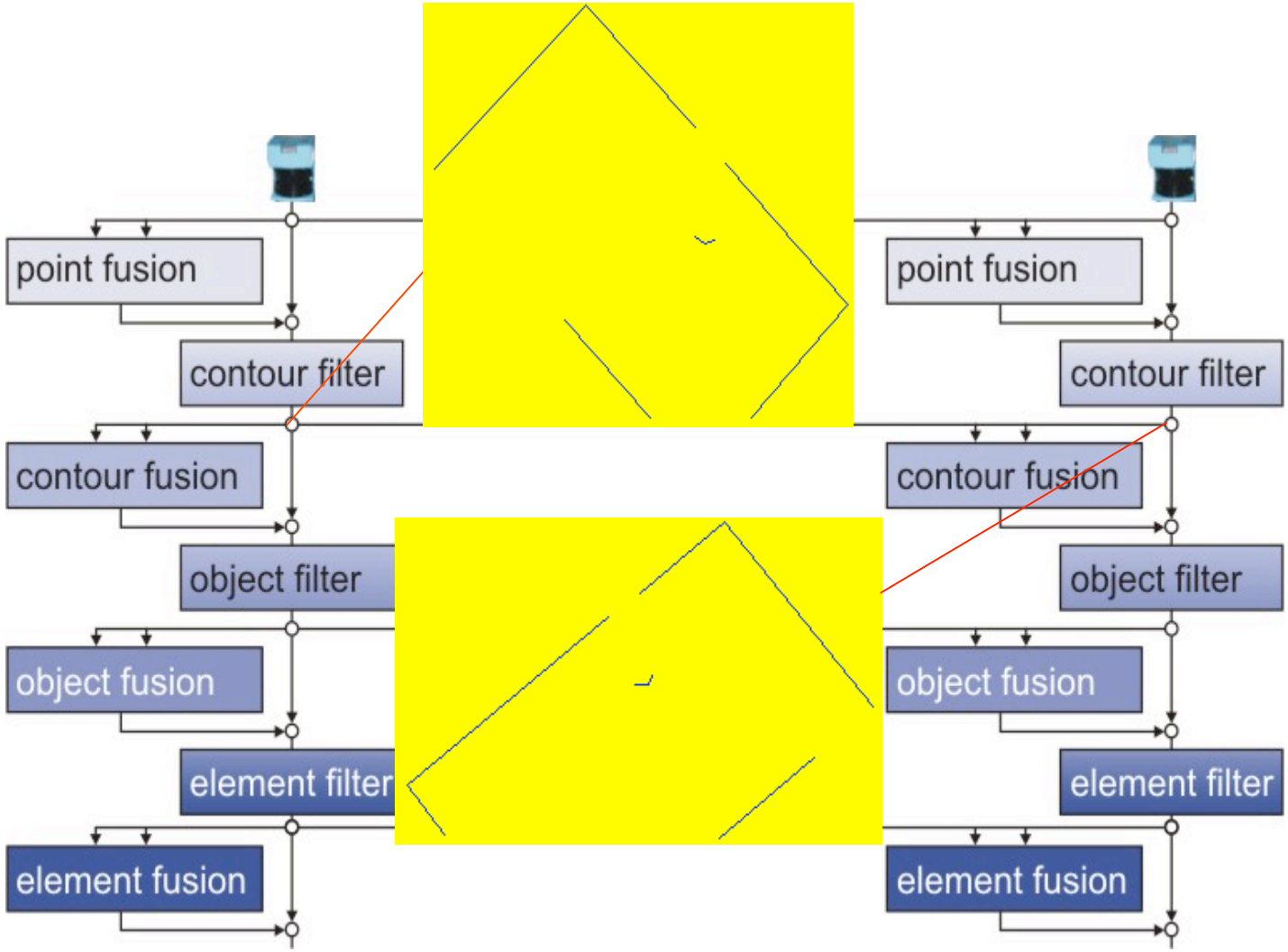
Application-Level Adaptation: Accuracy Tradeoff



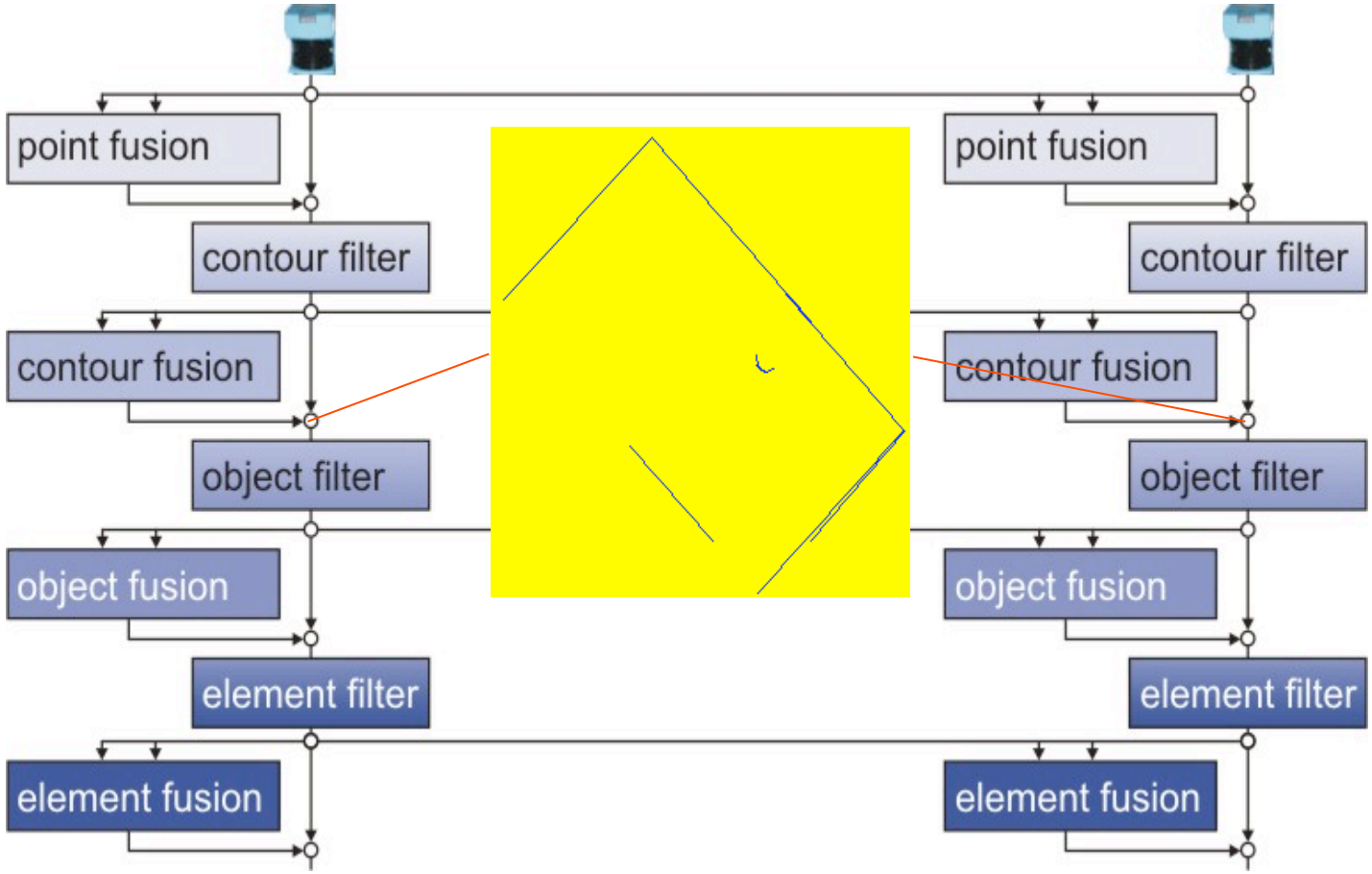
Application-Level Adaptation: Accuracy Tradeoff (cont'd)



Application-Level Adaptation: Accuracy Tradeoff (cont'd)



Application-Level Adaptation: Accuracy Tradeoff (cont'd)



Exploiting Inherent Redundancy

- To tolerate transient faults (short term peaks), use application-inherent redundancy
  - Functional redundancy within the module instances
    - → Design modules as any-time algorithms
  - Structural redundancy within the executions of the sensor fusion
    - → Combine results from distributed sensors
  - Time redundancy within a sequence of executions
    - → Schedule a frequency above the minimum
- To tolerate permanent faults (persistent overload), use application-level adaptation (graceful degradation)
  - TAFT supports detection of persistent overload
  - Application-level adaptation allows lowering system load