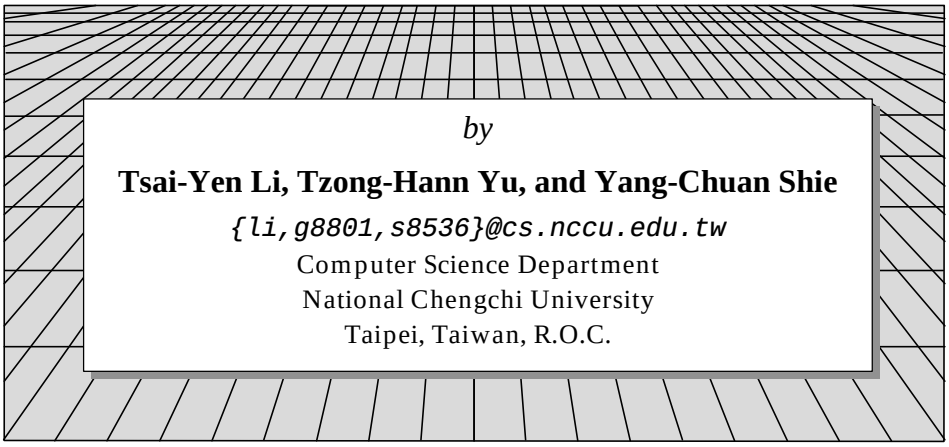
On-Line Planning for an Intelligent Observer in a Virtual Factory



by

Tsai-Yen Li, Tzong-Hann Yu, and Yang-Chuan Shie

{li,g8801,s8536}@cs.nccu.edu.tw

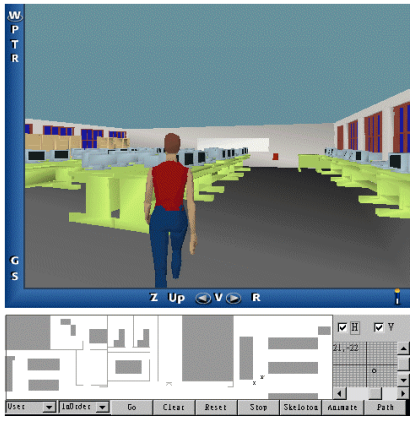
Computer Science Department
National Chengchi University
Taipei, Taiwan, R.O.C.

Outline

- **Introduction**
- **Problem formulation**
- **Search space and criteria**
- **The on-line planning problem**
- **Search space at run-time**
- **On-line planning algorithm**
- **Implementations and experiments**
- **Conclusion and future extensions**

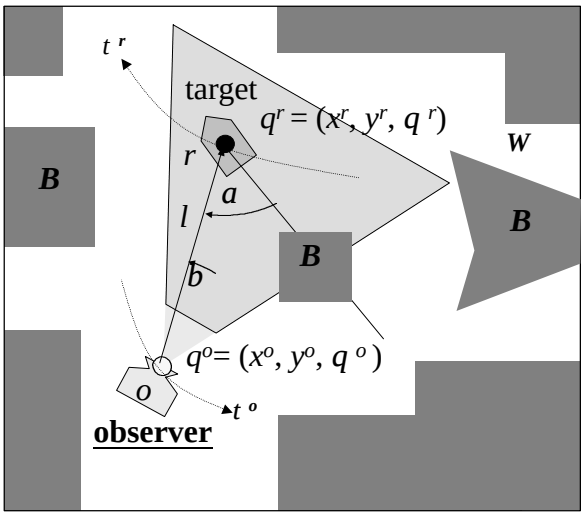
Introduction

- **An auto-navigation system for virtual environments:**
  - **specifying locations of interests by clicking on a 2D layout map**
- **The problems:**
  - **Tour path planning**
  - **Camera motion planning**
  - **Humanoid simulation**
- **Given a known target path, to generate an intelligent camera tracking motion to**
  - **avoid collisions with obstacles**
  - **always keep the target in sight**
  - **allow on-line interactive modification**



3

Problem Formulation: View Model



Target configuration:
$q^r = (x^r, y^r, q^r)$

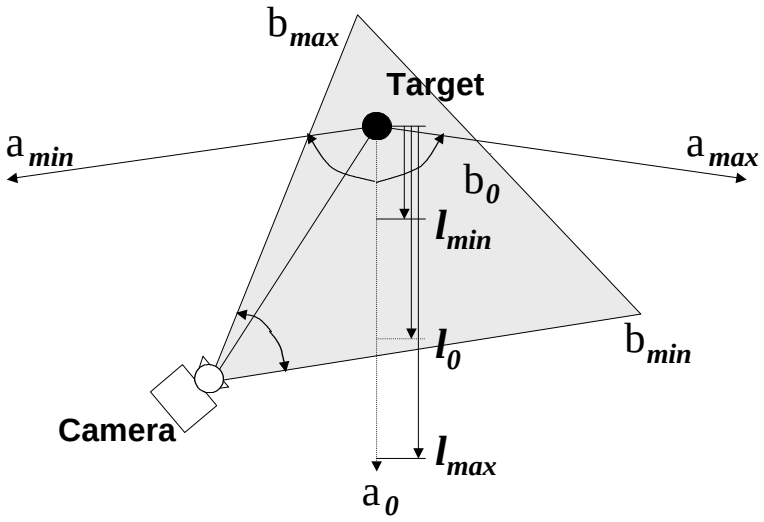
Viewpoint configuration:
$q^o = (x^o, y^o, q^o)$

Composite space:
$(x^r, y^r, q^r, x^o, y^o, q^o)$

Configuration-Time space (CT-space):
(t, x^o, y^o, q^o)

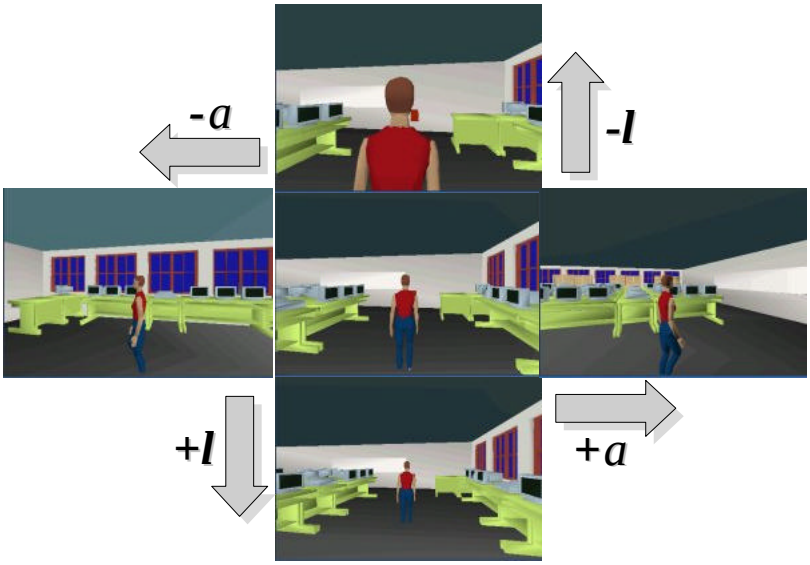
4

Problem Formulation: Planning Space Parameterization



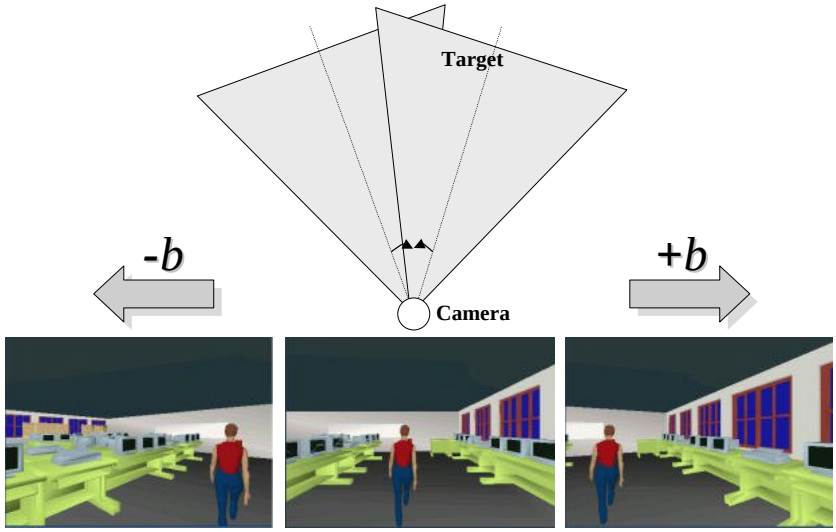
5

View Model Definitions: View Distance (*l*) and Tracking Direction (a)



6

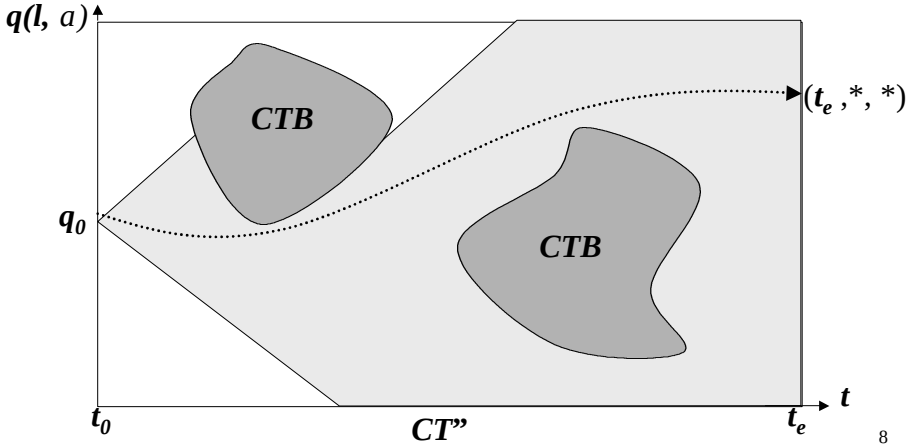
View Model Definitions: View Angle (b)



7

Search Space for the Planning Problem

- **Equivalent space:** CT (t, x^o, y^o, q^o) => CT' (t, a, l, b)
- **Simplification:** fixing view angle (b) => CT" (t, a, l)
- **Relaxing view angle** (b) **after a feasible path is found.**



8

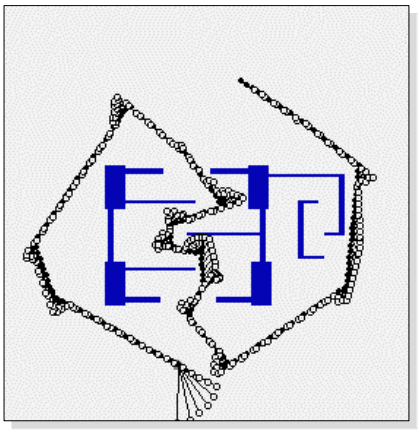
Search Criteria for Best-First Planning

- **A Best-First Planning (BFP) algorithm is used.**
- **The "*Best*" criteria used in the search:**
  - **Planning time (*t*):** highest priority
  - **Tracking Direction (*a*):** subjective criterion
  - **View Distance (*l*):** subjective criterion
  - **Overall Movement (*d*):** subjective criterion
  - **View Angle (*b*):** lazy movement in postprocessing
- **Cost function:**

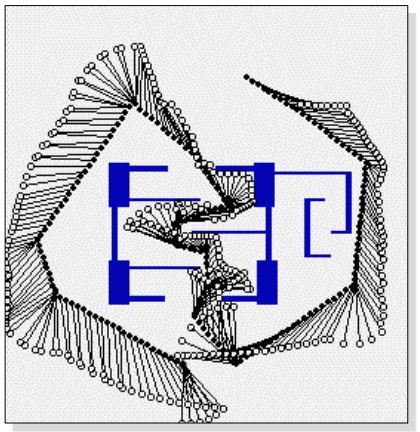
$$f(t, a, l, dir) = w_1 * f_1(\underline{t}) + w_2 * f_2(\underline{a}) + w_3 * f_3(\underline{l}) + w_4 * f_4(a, l, dir)$$

9

Examples of Observer's Motions



Prefer Good Tracking Direction (*a*)



Prefer Good View Distance (*l*)

○ **observer** ● **target**

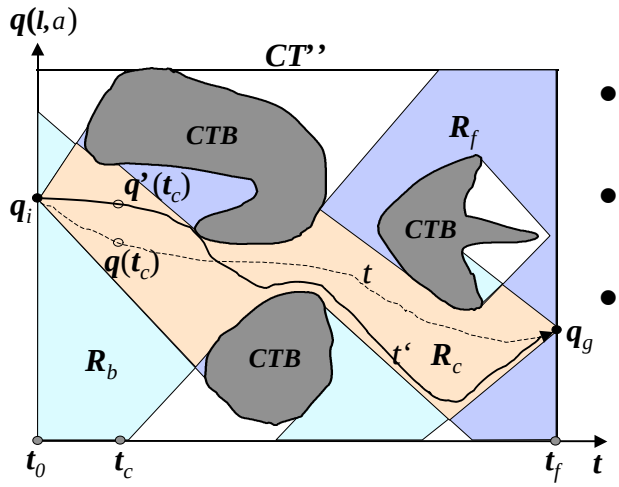
10

Motivation for On-Line Planning

- **Problem:**
  - **The choice of search criterion is subjective. The user may not be satisfied with the planner-generated paths.**
- **Solution:**
  - **We would like to develop an on-line algorithm to allow the user to modify the planned path interactively in an on-line manner**
  - **The visibility constraint is guaranteed to be satisfied.**

11

Search Space for Maintaining Visibility at Run-Time



- R_f **:** forward reachable region
- R_b **:** backward projection region
- R_c **:** cross section of R_f and R_b in *CT"*, legal region for transforming t.

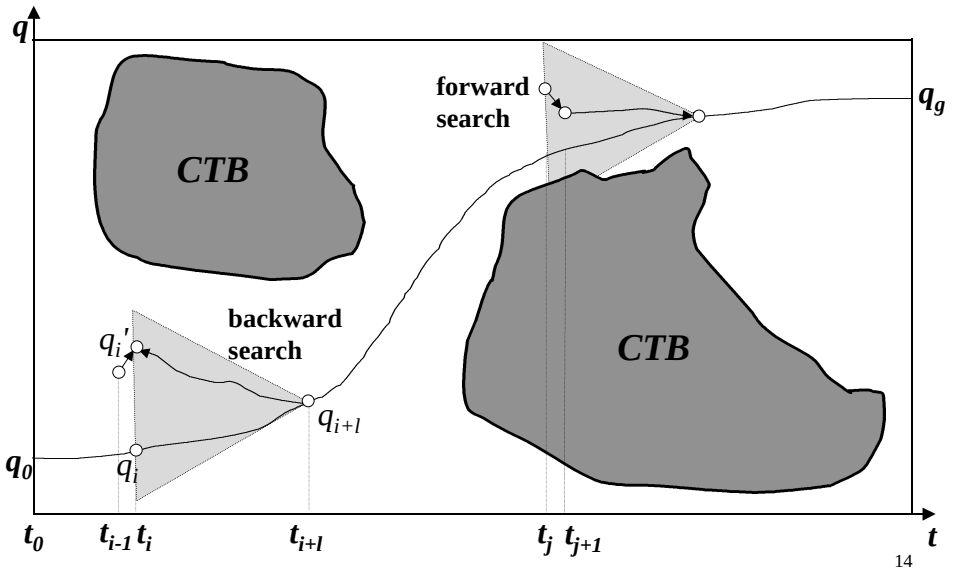
12

Ideas for the On-line Planning Algorithm

- **One can not afford to search the whole CT-space on-line in each frame update to ensure visibility.**
- **Propose to use an incremental search algorithm.**
  - **At each time step, searched regions in earlier steps do not need to be searched again.**
- **Need to distinguish two cases:**
  - ***Backward search*: when the desired configuration has NOT been visited.**
  - ***Forward search*: when the desired configuration has been visited.**

13

Example of On-Line Planning



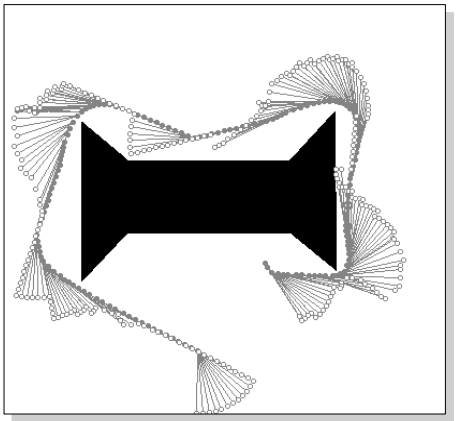
14

Implementations and Experiments

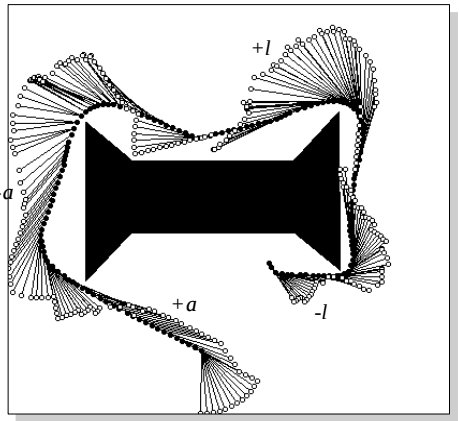
- **Implementation:** The on-line planner is written in Java.
- **Collision detection:** a key module for planning efficiency
  - **Use a 3D distance map to speed up visibility checks.**
  - **The value in each cell of this map represents the distance to the first obstacle boundary from the corresponding configuration.**
- **Experimental settings:**
  - **Planning time is measured on a Celeron 400 MHz PC.**
  - **Workspace: 128x128 grid**
  - **Rotational increment: 3 degrees**
- **Planning performance:**
  - **Average number of configurations visited for each frame: 40.**
  - **Maximal planning time in an execution step is around 250ms.**

15

Experimental Results: An Example of On-Line Modification



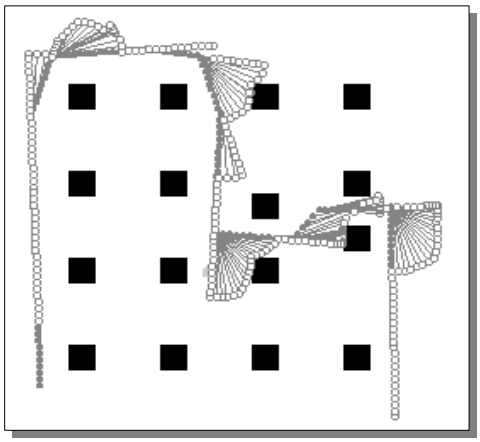
Planned path



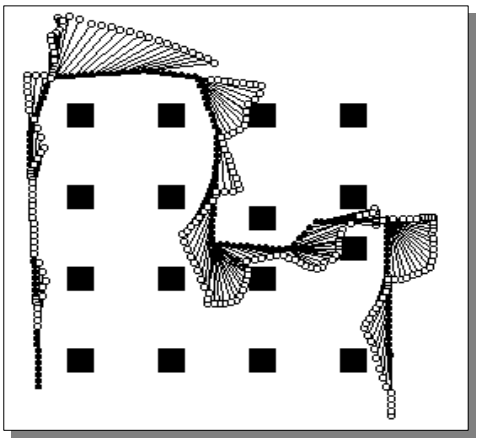
On-line modified path

16

Experimental Results: Another Example of On-Line Modification



Planned path

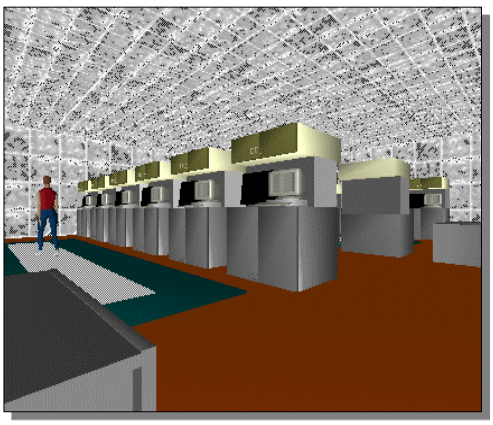


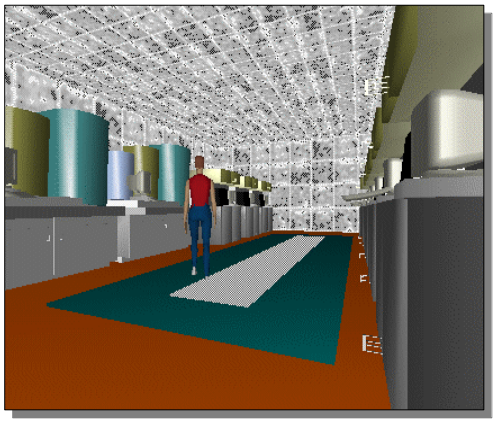
On-line modified path

17

A Virtual Factory Example: 3D Graphical User Interface

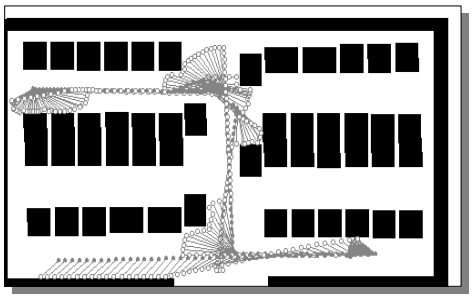
3D Virtual Factory Model



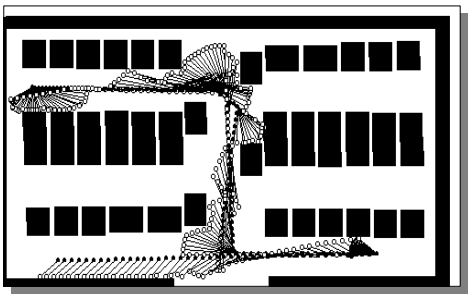


18

A Virtual Factory Example: On-Line Planning Example



Planned path



On-line modified path

19

Conclusion and Future Extensions

- **Proposing a planning approach for tracking moving target with an intelligent virtual camera**
  - **Finding a good path quickly for interactive applications**
  - **Proposing an incremental searching algorithm for on-line modification of the observer's motions**
- **Future Extensions:**
  - **predicting the target's motion which is unknown in advance.**
  - **handling 3D environments with full camera motions.**
  - **integrating the planner into a virtual presence system, connected to 3D user interface, such as VRML browser.**

20