Score Detection and Strategy Optimization in the game of Darts Using Frame Difference and Dynamic Programming

B3 Sutashu Tomonaga

Outline

- Background and Objective
- Method
- Results

Outline

- Objective and Background
- Methodology
- Conclusion

Background

- Motivation
  - Kept on losing when playing darts with friends and was made fun of
  - Felt obligated to win at any cost
  - I shall win by optimization

Background

- Problem
  - My lack of skill can not be ignored
  - However, the game of Darts is not only about skill; strategy plays an important roll
  - Can I formulate an optimized strategy taking into account the momentary condition and habits of the player?

Objective

- Find optimal Strategy for the game of darts considering the remaining score and the players condition.

Game rules

- 301 game
  - Players start with 301 points, subtracting the attained score from the remaining score after every throw
  - Whoever reaches the score 0 first wins.

Game rules

- Example of a game

Remaining Score	301	→	281	→	…	→	19	→	5	→	1	→	0
		∧						∧		∧		∧	
Attained Score		20			…			14		4		1	

- In other words, the objective is to have the least number of steps to reach 0

Outline

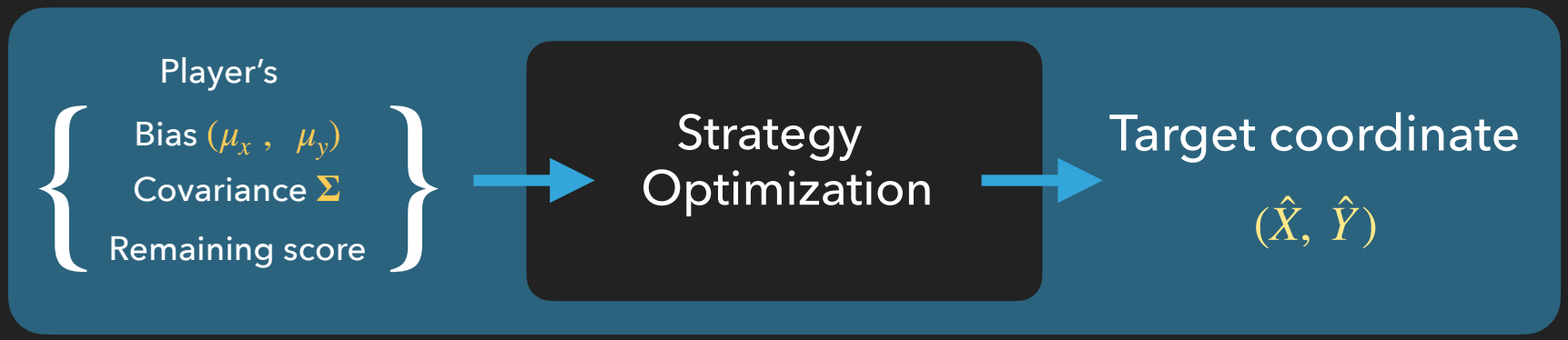
- Objective and Background
- Methodology
- Conclusion

How to define a player's "condition" or "habits"

Assuming we have a record of the coordinates of where the player aimed and where the dart landed:

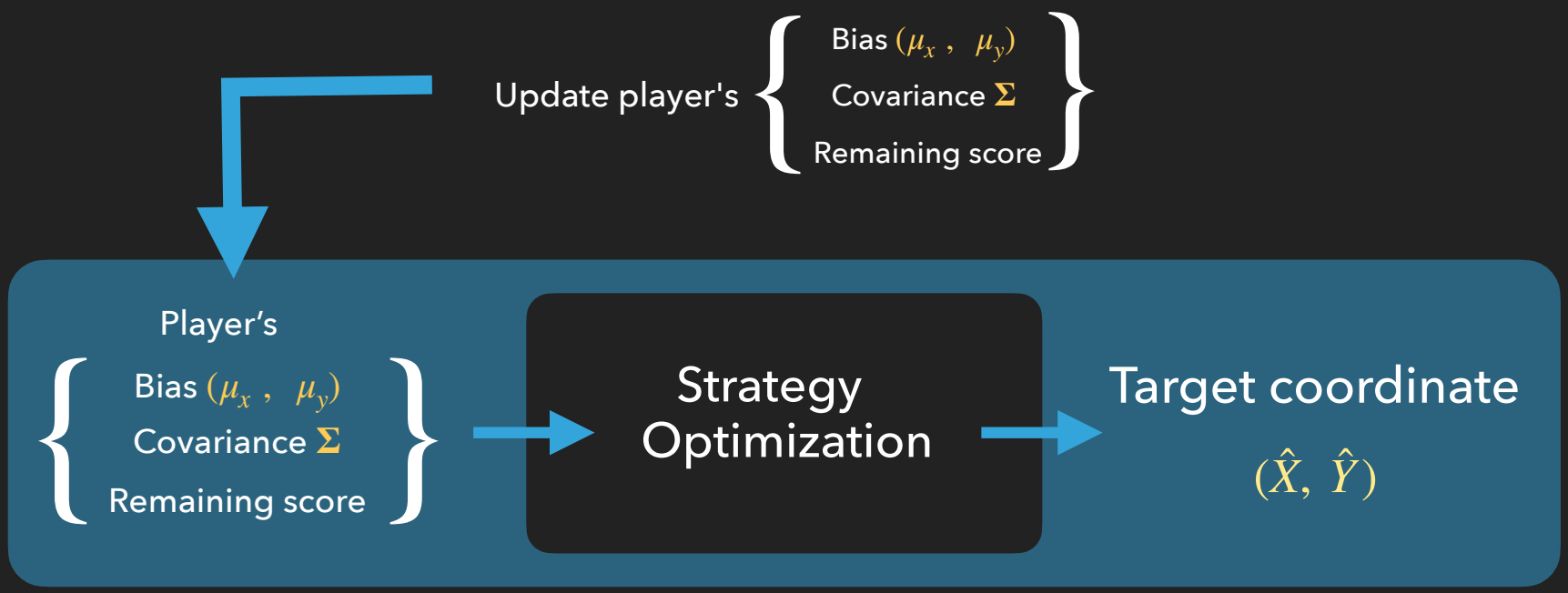
- We may evaluate the "condition" of the player by calculating the bias $(\mu_x\ ,\ \ \mu_y)$, and the covariance Σ based on past data recorded.

System Configuration



- Condition: Defined as (bias $(\mu_x\ ,\ \mu_y)$, covariance Σ), and will be updated sequentially

System Configuration



System Configuration

Input image

Detect landing coordinate

Landing coordinate (x, y)

Attained score

Update player's

Bias $(\mu_x\ ,\ \mu_y)$

Covariance Σ

Remaining score

Player's

Bias $(\mu_x\ ,\ \mu_y)$

Covariance Σ

Remaining score

Strategy Optimization

Target coordinate

$(\hat{X},\ \hat{Y})$

System Configuration

Input image

Detect landing coordinate

Landing coordinate (x, y)

Attained score

Update player's

Bias $(\mu_x\ ,\ \mu_y)$

Covariance Σ

Remaining score

Player's

Bias $(\mu_x\ ,\ \mu_y)$

Covariance Σ

Remaining score

Strategy Optimization

Target coordinate

$(\hat{X},\ \hat{Y})$

System Configuration

Input image

Detect landing coordinate

{ Landing coordinate (x, y) Attained score }

{ }

{ }

$(\hat{X}, \hat{Y})$

Detecting Landing Coordinate

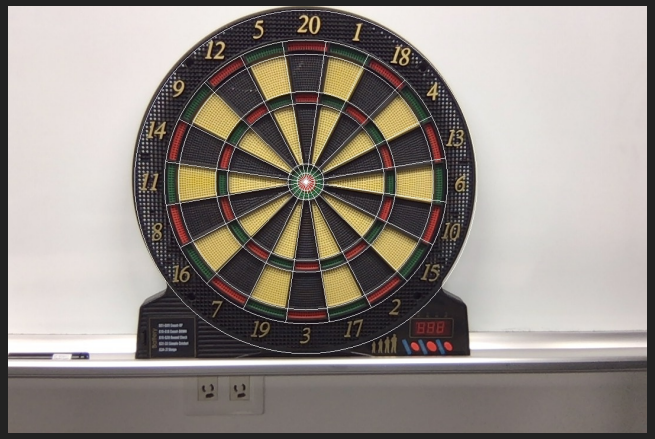
Input

- Reference image
- Background image
- Image with thrown dart

output

- Landing coordinate (x, y)
- Attained score

Input

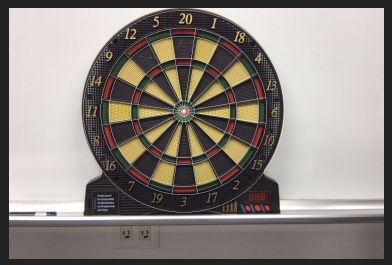


Reference image

Background image

Image with thrown dart

Input



Reference image

Image with thrown dart

Background image

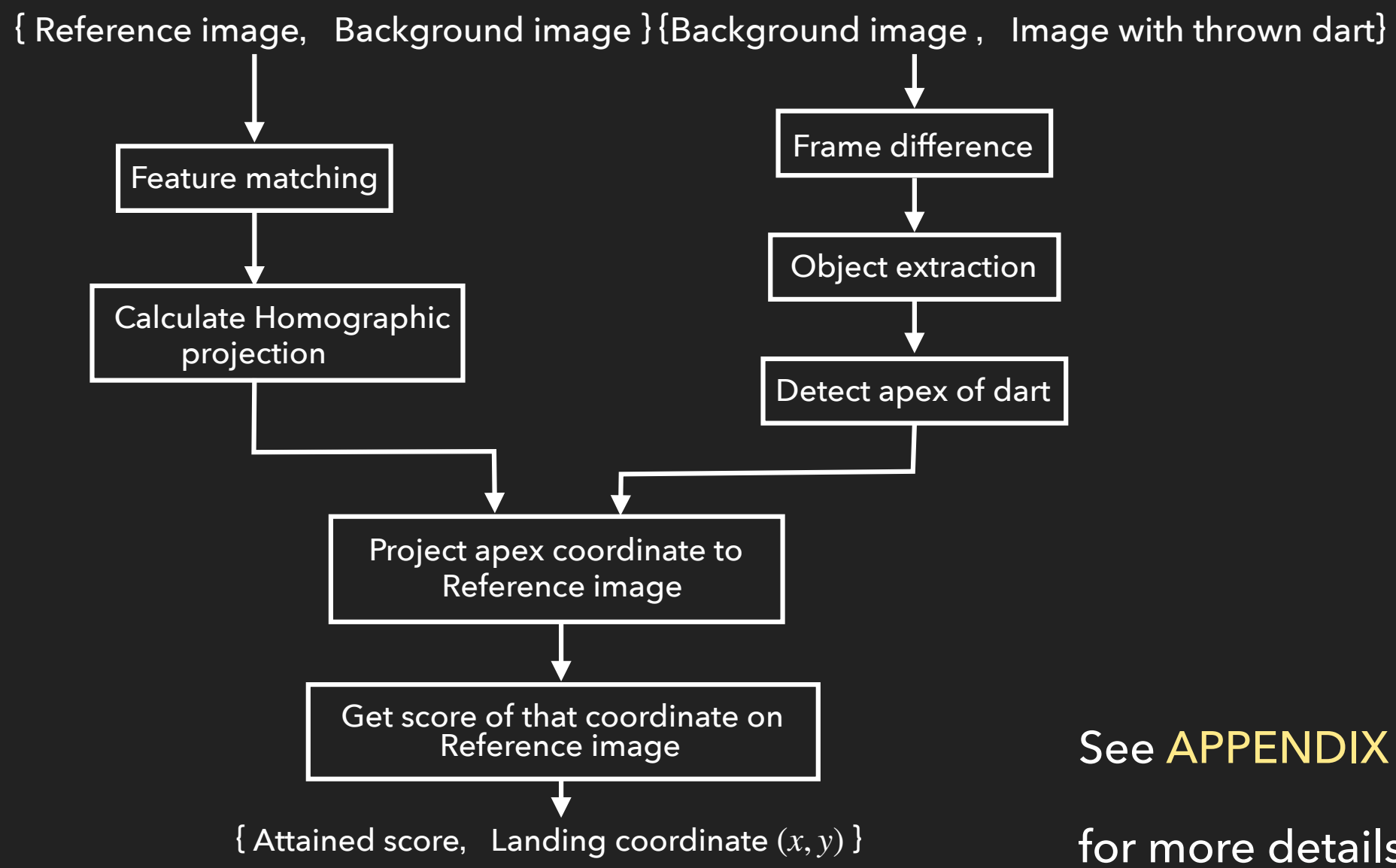
output

Landing coordinate

(x, y)

Attained score

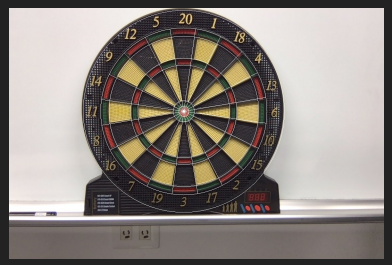
Detection flow



See APPENDIX

for more details

Input



Reference image

Image with thrown dart

Background image

output

$$\left\{ \begin{array}{r} \text{Landing coordinate } (x, y) \\ \text{Attained score} \end{array} \right\}$$

System Configuration

Input image

Detect landing coordinate

Landing coordinate (x, y)

Attained score

Player's

Bias (μ_x, μ_y)

Covariance Σ

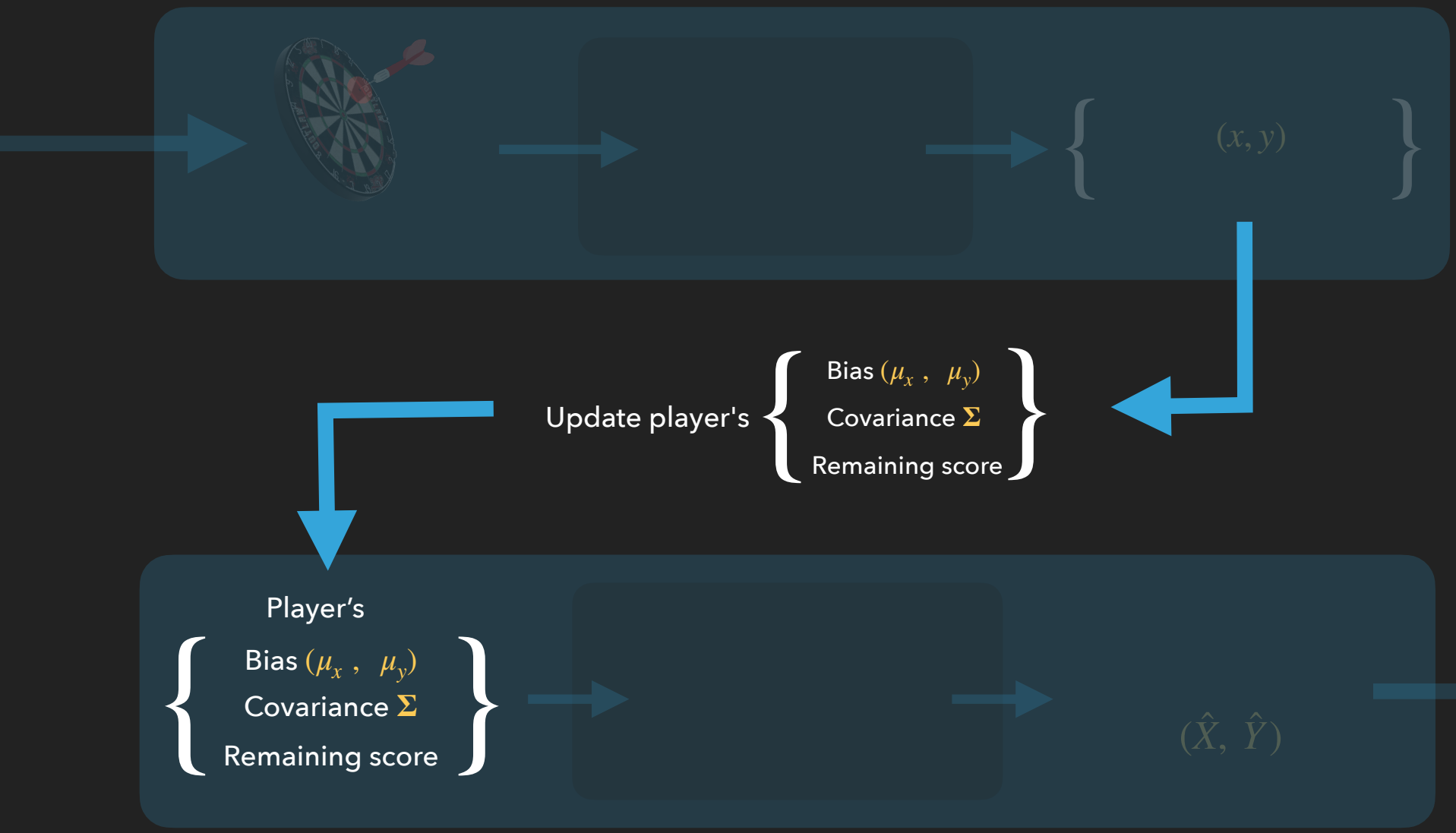
Remaining score

Strategy Optimization

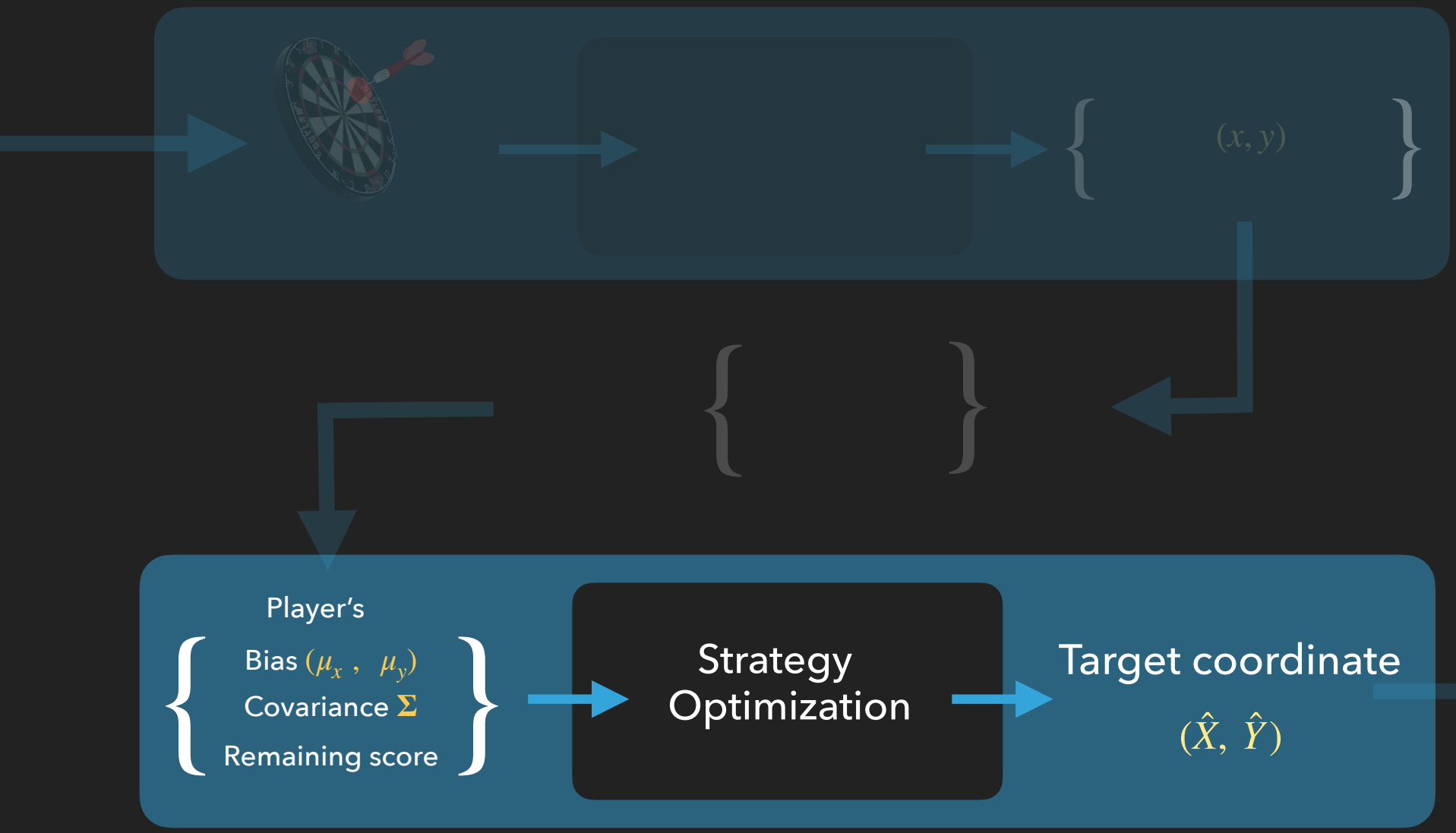
Target coordinate

$(\hat{X}, \hat{Y})$

System Configuration



System Configuration



Input-Output Strategy Optimization

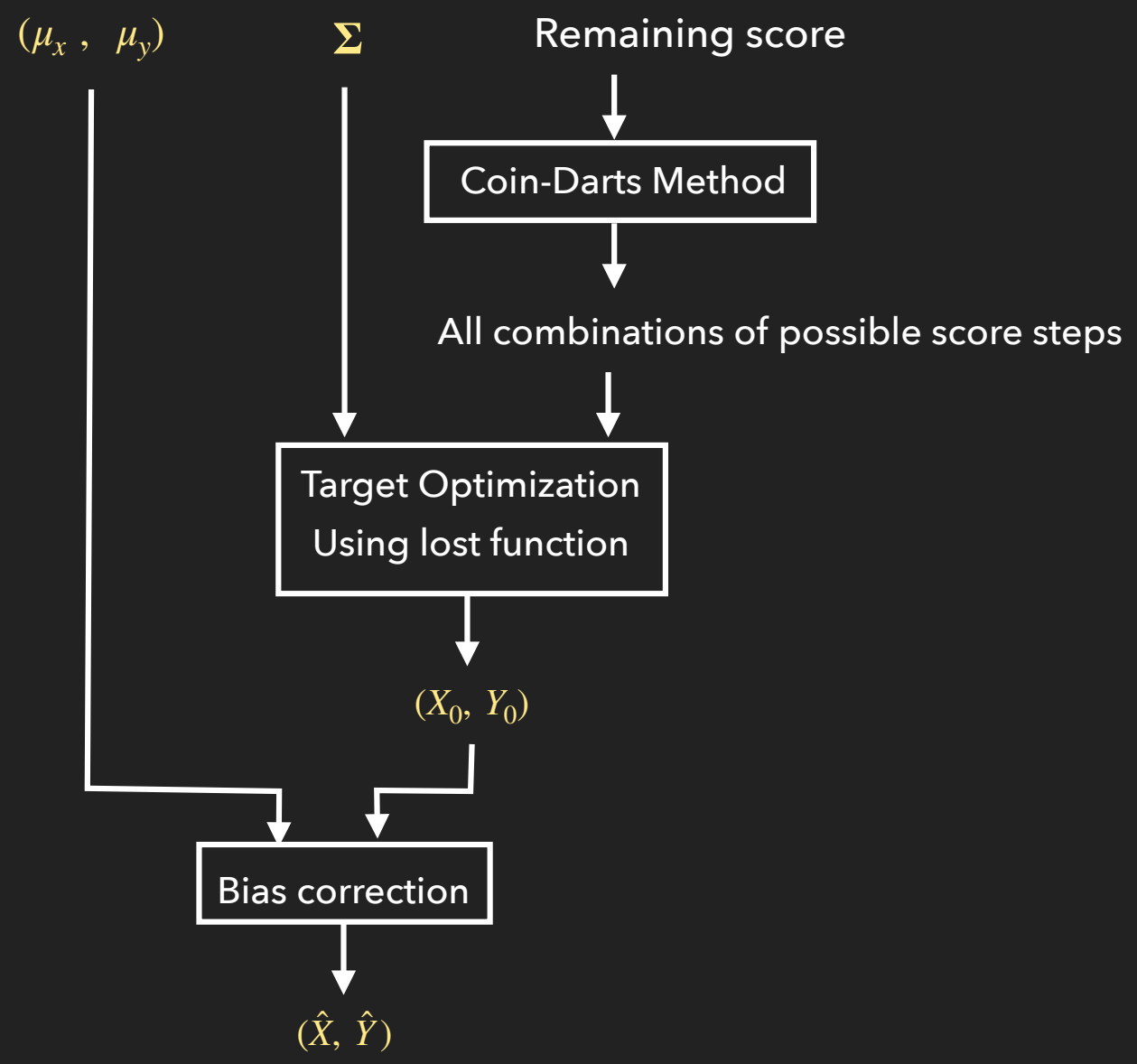
Input

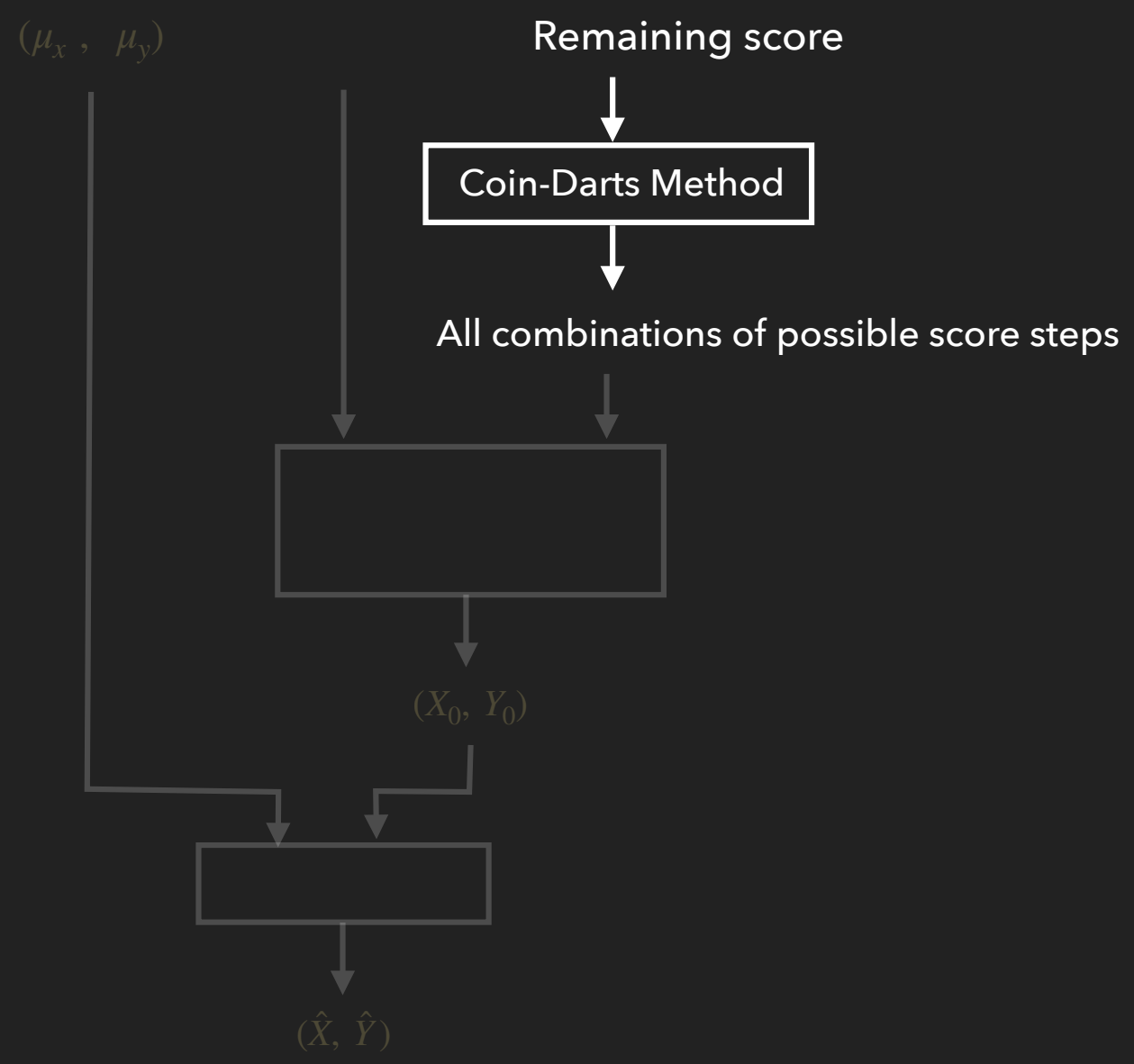
- Player's bias
  $(\mu_x\ ,\ \ \mu_y)$
- Player's covariance
  $\boldsymbol{\Sigma}$
- Remaining score

output

- Target coordinate
  $(\hat{X},\ \hat{Y})$

Detectioon flow




$(\mu_x\ ,\ \ \mu_y)$
Remaining score
Coin-Darts Method
All combinations of possible score steps
$(X_0,\ Y_0)$
$(\hat{X},\ \hat{Y})$

How to optimize

Remaining score	301	→	281	→	…	→	19	→	5	→	1	→	0
	∧						∧		∧		∧		
Attained score	20				…		14		4		1		

- The objective is to have the least number of steps to reach 0

How to optimize

Aim

- Look for the combination of scores with the least number of steps

Remaining score	301	→	281	→	…	→	19	→	5	→	1	→	0
		∧						∧		∧		∧	
Attained score		20			…			14		4		1	

How to optimize

Similar problem：the Coin Change Problem

Given m numbers of coins: $c_1, c_2, \ldots, c_m$, find the combination with the least number of coins to make up the number n

- Dynamic Programming lets us calculate and store solutions for any given m, n.

How to optimize

Why use dynamic Programming?

- Intuitively, it seems best to start with the largest coin → **not optimal**
  - We must consider the combinations of the remaining amount
  - Example:
    - When given (1, 3, 7, 8, 11) coins, and n =15
      - Solution if payed first with 11: (11, 3, 1) or (11, 1, 1, 1, 1)
      - Optimal solution: (7, 8)

How to optimize

Coin Change Problem	When applied to Darts
• m different coins: $c_1, c_2, \ldots, c_m$	• $m = 62$ number of different scores on the dart board
• Combination of least number of coins when paying the price n	• Combination of scores with the least number of steps to achieve $n = 0$

Reestablishing the problem

Coin-Darts Method

- Using the $m = 62$ different scores on the dart board $c_1, c_2, \ldots, c_m$, find the combinations of scores with the least number of steps to achieve $n = 0$.

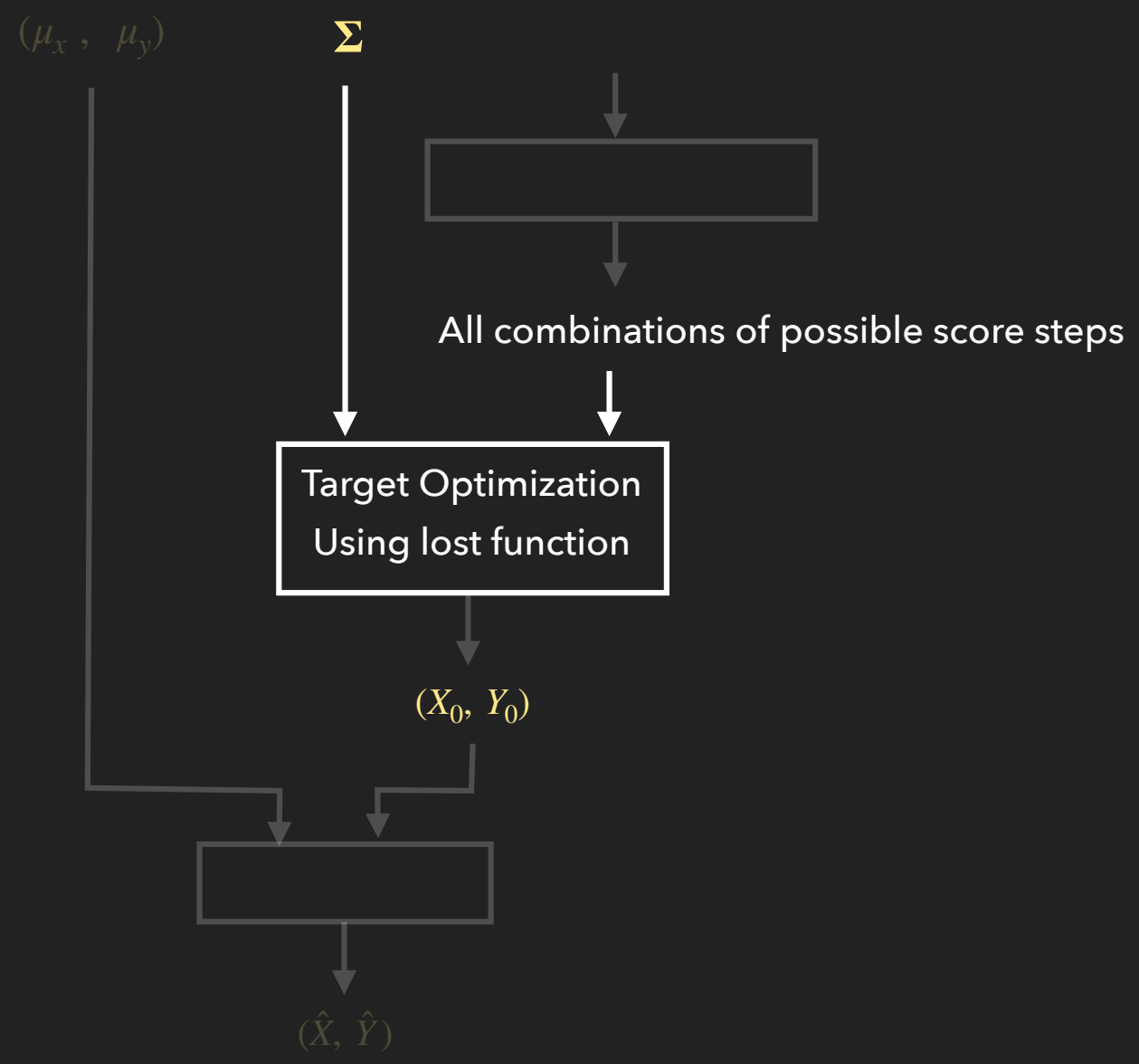
  - More specifically, for each scores $c_1, c_2, \ldots, c_m$, find the minimal combination containing c_i that achieves $n = 0$

Reestablishing the problem

Coin-Darts Method

- Using the $m = 62$ different scores on the dart board $c_1, c_2, \ldots, c_m$, find the combinations of scores with the least number of steps to achieve $n = 0$.

- Example
  - Remaining score = 6
    - result = [(6,), (4, 2), (5, 1), (3, 3)]


$(\mu_x\ ,\ \mu_y)$
Σ
All combinations of possible score steps
Target Optimization
Using lost function
$(X_0,\ Y_0)$
$(\hat{X},\ \hat{Y})$

How to utilize results from Coin-Darts Method

- We want to consider the risk of missing when aiming for a combination of scores
  - Define a loss function L where the loss is the sum of risk for missing a given target coordinate
  - Risk R defined as the total number of steps remaining for each coordinate when landed upon

How to utilize results from Coin-Darts Method

- Defining risk R:

$$R\,(c_i \mid (X,\ Y))$$

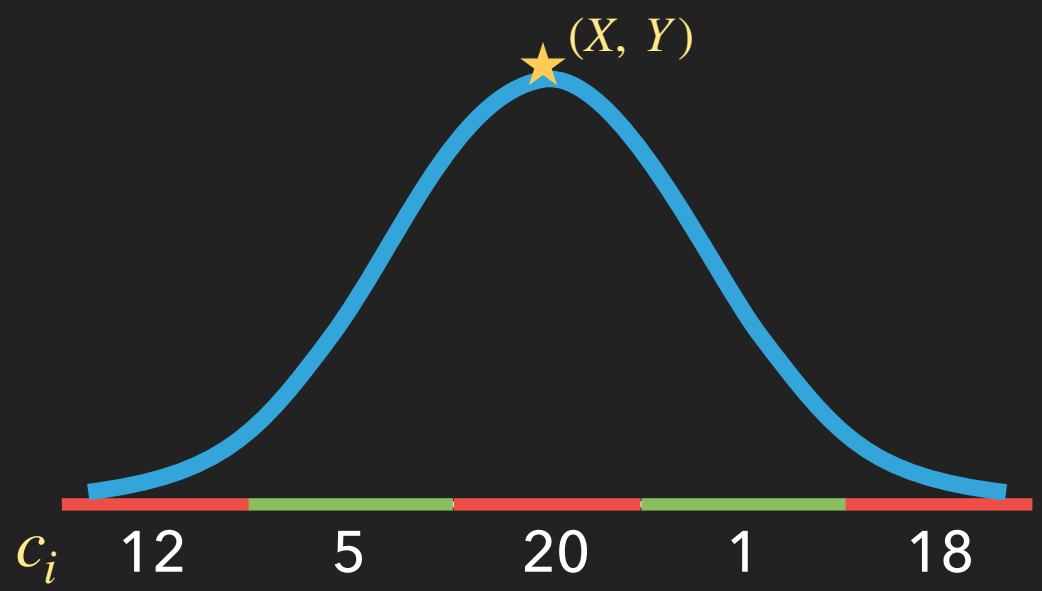
- Total number of steps remaining, when aiming at target $(X,\ Y)$, and landed on score region c_i

Target Optimization Using lost function

- Defining lost function L:
  - The average risk for aiming at coordinate (X, Y)

Target Optimization Using lost function

- Defining lost function L:
  - The average risk for aiming at coordinate $(X,\ Y)$



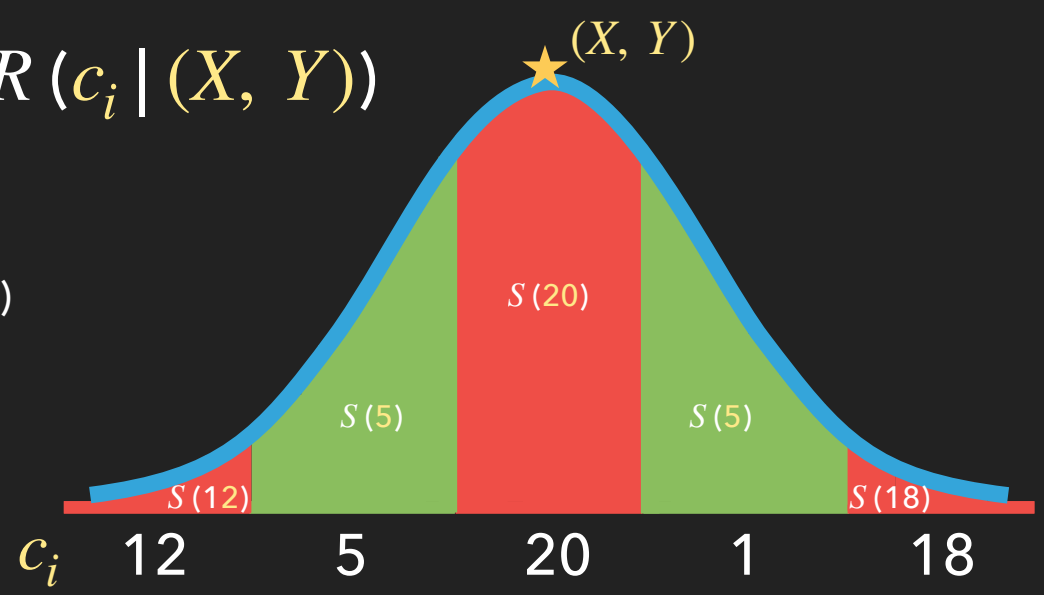
Assumed probability distribution when aiming for $(X,\ Y)$

Target Optimization Using lost function

- Defining lost function L:
  - The average risk for aiming at coordinate $(X,\ Y)$

$$L(X,\ Y) = \sum_{i=0}^{m} S(c_i \mid (X,\ Y))\, R(c_i \mid (X,\ Y))$$

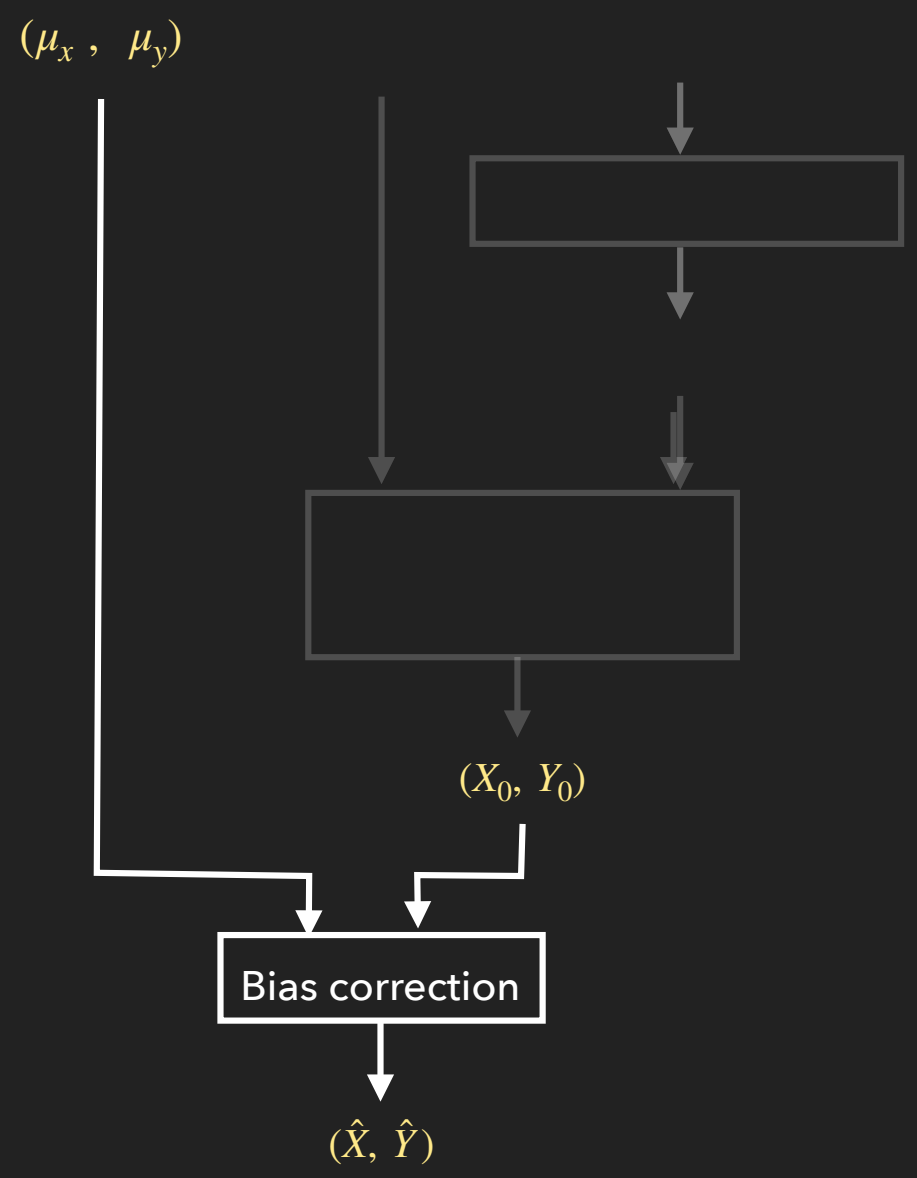
$$(R(1) > R(5) > R(12) > R(18) > R(20))$$



Target Optimization Using lost function

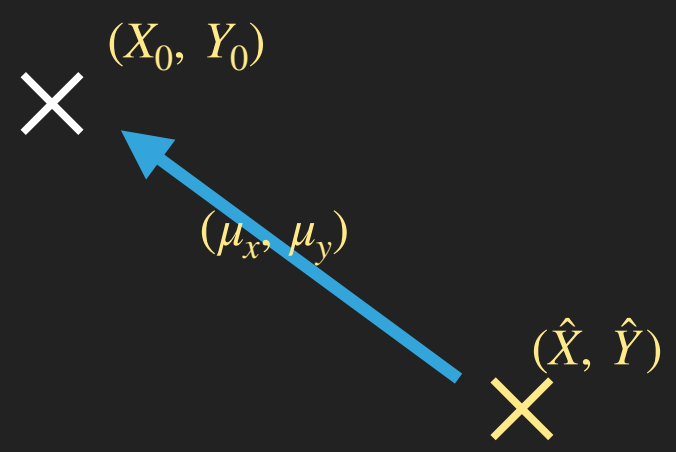
- Defining lost function L:
  - The average risk for aiming at coordinate (X, Y)

- Pros
  - Able to evaluate the strategic risk of missing a target
- Cons
  - Requires time to optimize


$(\mu_x\ ,\ \ \mu_y)$
$(X_0,\ Y_0)$
Bias correction
$(\hat{X},\ \hat{Y})$

Bias correction

- Correct target $(X_0,\ Y_0)$ using bias vector $(\mu_x,\ \mu_y)$
  - $(\hat{X},\ \hat{Y}) = (X_0,\ Y_0) - (\mu_x,\ \mu_y)$



Outline

- Results

output results

- $n = 6, \quad (\mu_x \ , \ \mu_y) = [-5, -5], \quad \Sigma = [[10, 0], [0, 8]]$
  - Before bias correction
  - After bias correction

Green: Optimal target, Red: Small loss (good target), Yellow: Large loss (bad target)

output results

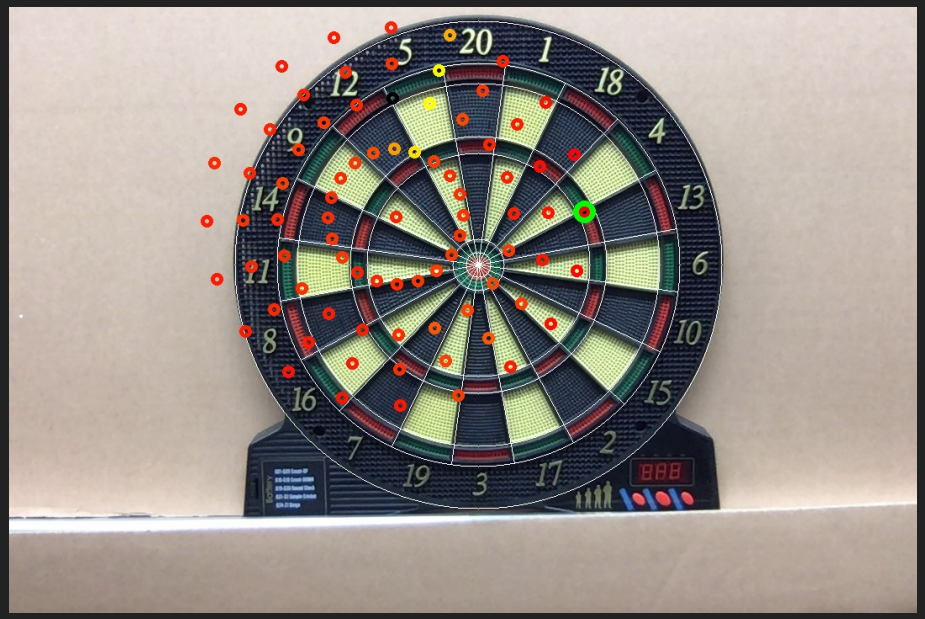
- $n = 23, \quad (\mu_x \ , \ \mu_y) = [-5, -5], \quad \Sigma = [[10, 3], [3, 8]]$
  - Before bias correction
  - After bias correction

Green: Optimal target, Red: Small loss (good target), Yellow: Large loss (bad target)

T = 1:23

output results

- $n = 57, \quad (\mu_x \ , \ \ \mu_y) = [-5, -5], \quad \Sigma = [[10, 3], [3, 8]]$
  - Before bias correction
  - After bias correction



Green: Optimal target, Red: Small loss (good target), Yellow: Large loss (bad target)

T = 5:14

Summary

- Objective：Find optimal Strategy for the game of darts considering the remaining score and the players condition.
- Methodology：

1. Detecting the Landing Coordinate
   - Input：Reference image, Background image, Image with thrown dart
   - output：Attained score, Landing coordinate (x, y)
   - Method：Object recognition using frame difference
2. Update player's bias $(\mu_x\ ,\ \mu_y)$ and covariance $\mathbf{\Sigma}$
3. Strategy Optimization
   - Input：Player's bias $(\mu_x\ ,\ \mu_y)$, Player's covariance $\mathbf{\Sigma}$, Remaining score
   - output：Target coordinate $(\hat{X},\ \hat{Y})$
   - Method：Dynamic Programming, Lost function minimization, and bias correction

謝辞

- 問題を一緒に考えてくださった青田さん，楯さん，奥田くん

APPENDIX

Defining the Lost Function

When S is the set of all combinations results for the Coin-Darts Problem and $K = |S|$ is the number of combinations

$$S = \{S_1, \cdots, S_k, \cdots, S_N\}, K = |S|$$

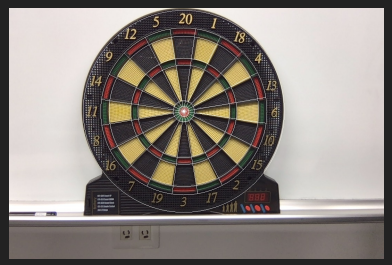
S_k is the set of steps for each combination and $M_k = |S_k|$ is the number of steps

$$S_k = \{S_{k1}, \cdots S_{km}, \cdots, S_{kM_k}\}, M_k = |S_k|$$

We define the optimal solution as follows

$$(\hat{X}, \hat{Y}) = \underset{(X,Y)}{\operatorname{argmin}}\left(\sum_{k=1}^{K} M_k \sum_{m=1}^{M_k} p\left(S_{km} | (X, Y)\right)\right)$$

Input



Reference image

Image with thrown dart

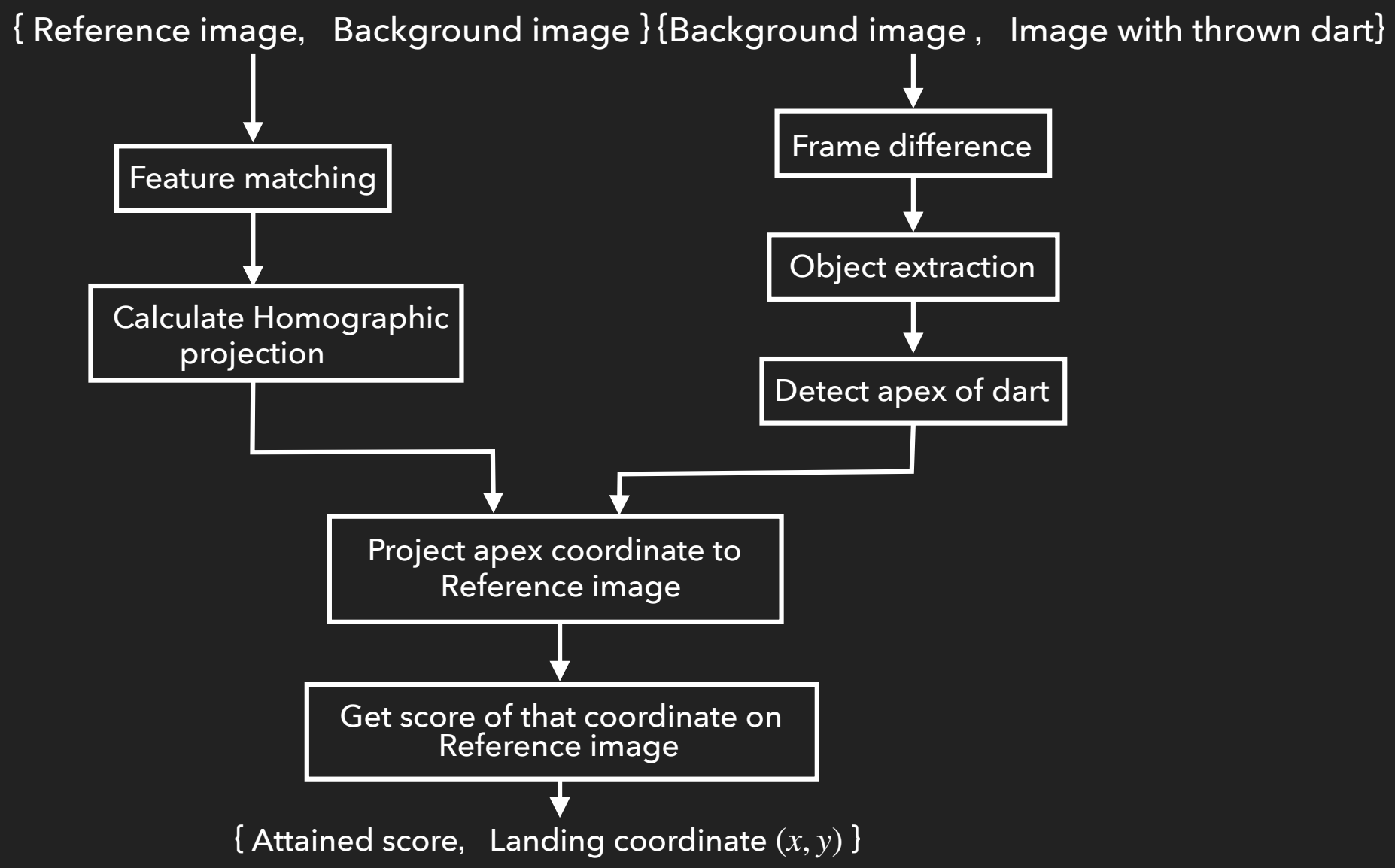
Background image

output

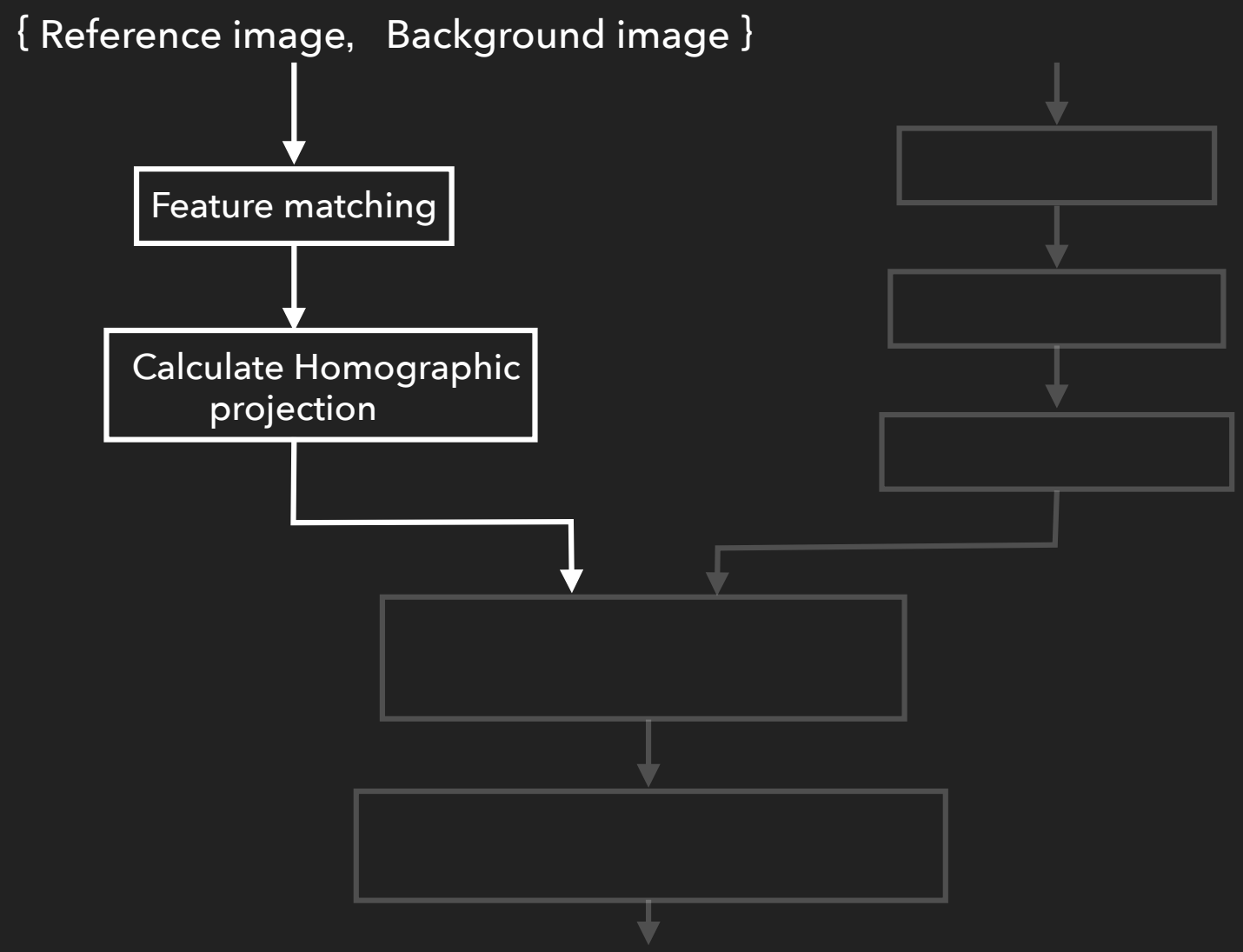
Landing coordinate

Attained score

Detectioon flow



Detectioon flow



Feature matching and calculating the projection

{ Reference image, Background image }

↓

Feature Matching

↓

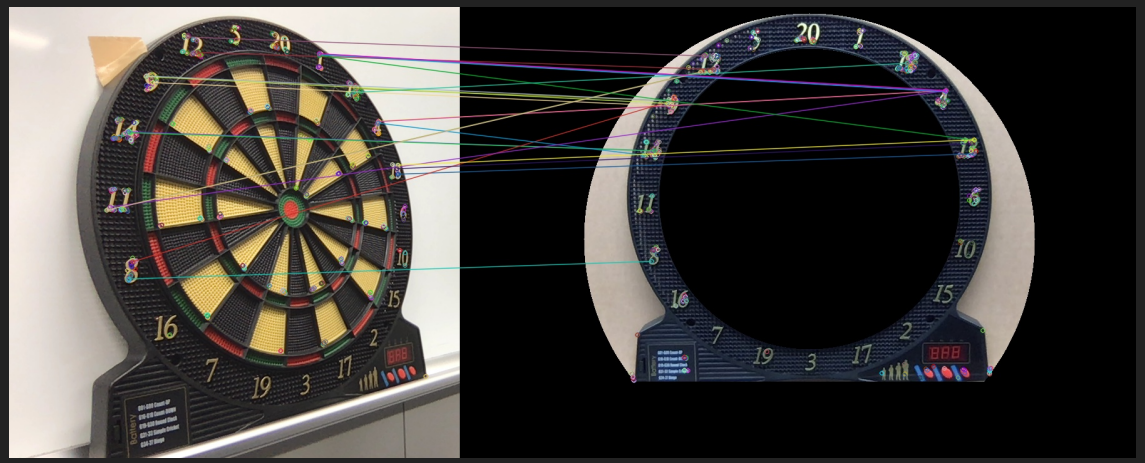
{ feature points }

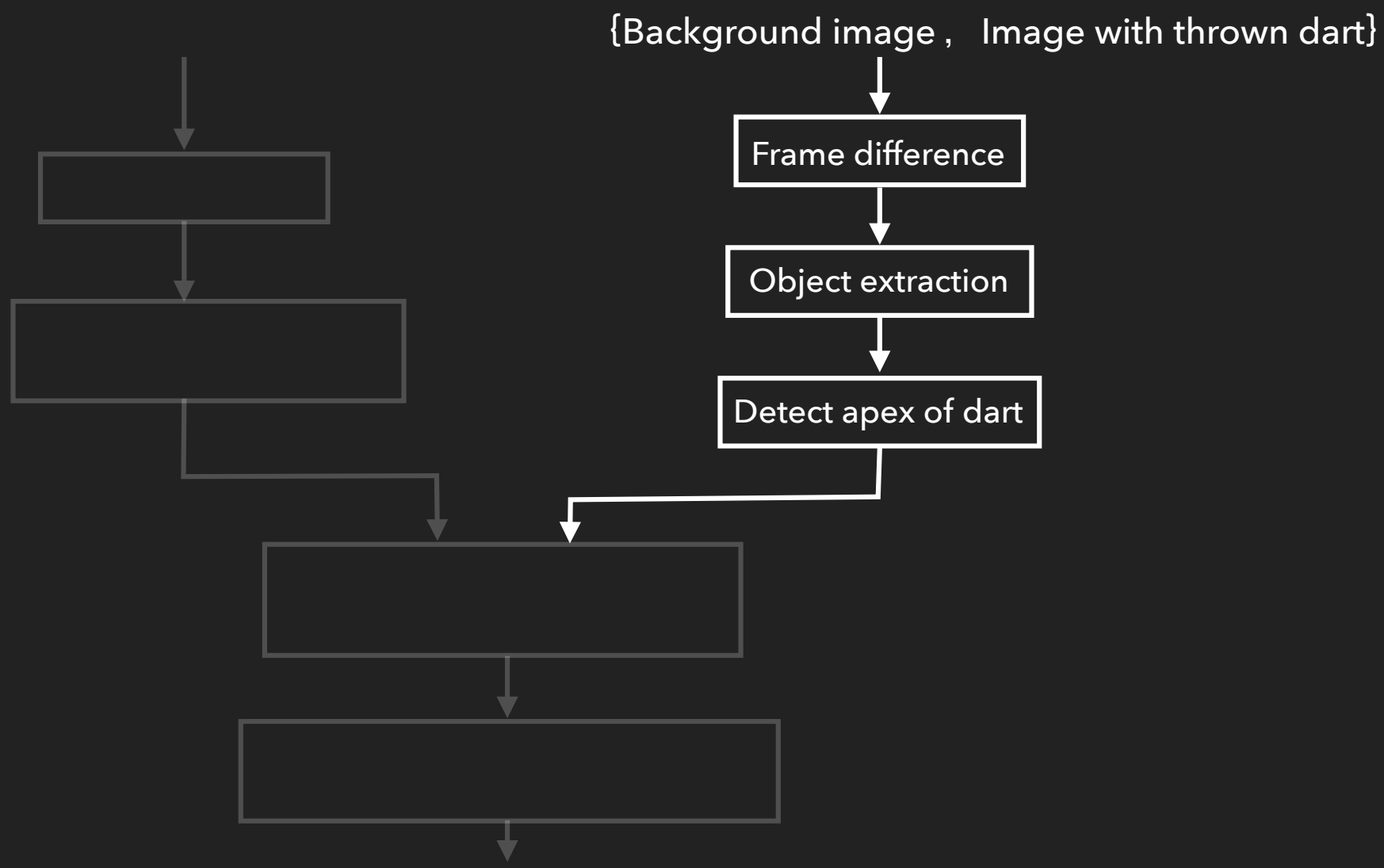
↓

Calculate projection

↓

Homographic projection matrix H




{Background image , Image with thrown dart}
Frame difference
Object extraction
Detect apex of dart

Detecting dart's apex from frame difference and object extraction

{Background image , Image with thrown dart}

↓

Frame difference

↓

Difference image

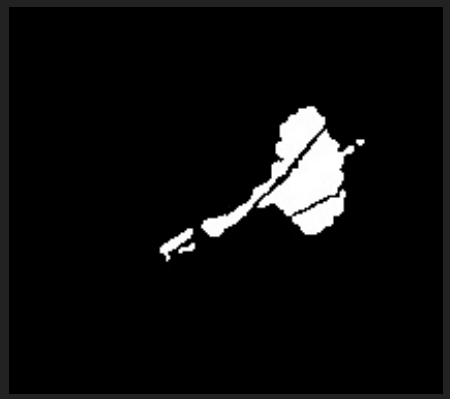
↓

Object extraction

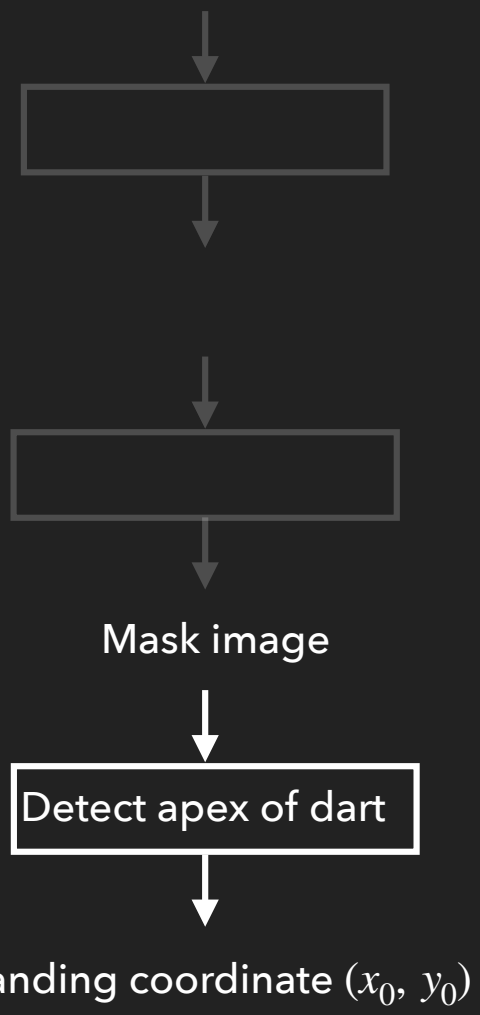
↓

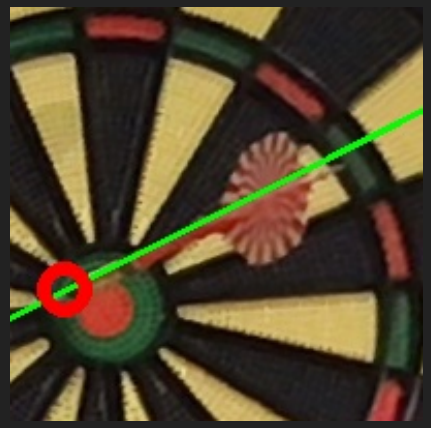
Mask image

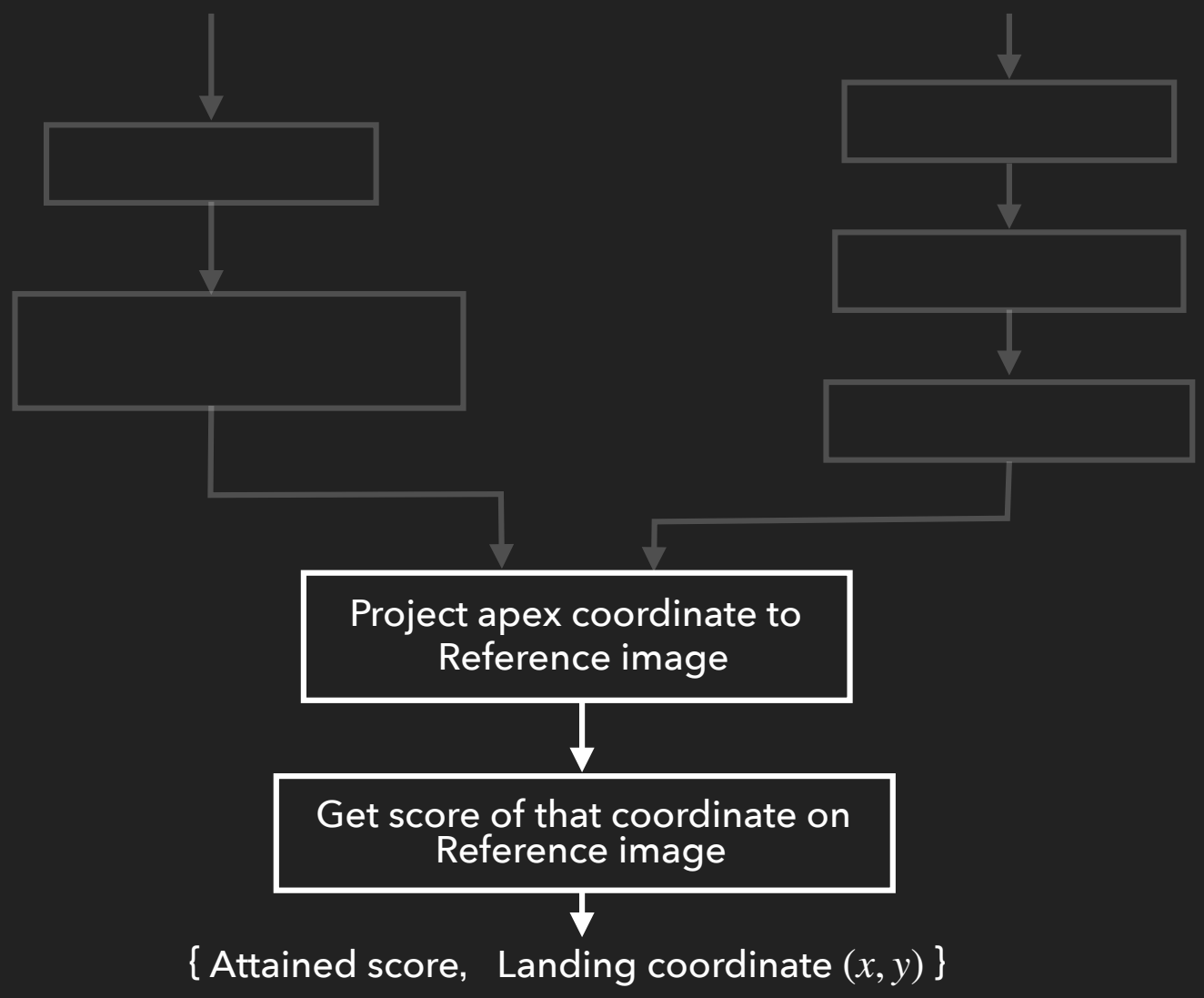
↓



Detecting dart's apex from frame difference and object extraction






Project apex coordinate to Reference image
Get score of that coordinate on Reference image
{ Attained score, Landing coordinate (x, y) }

Finding the projection matrix and the attained score

Homographic projection matrix H

Landing coordinate (x_0, y_0)

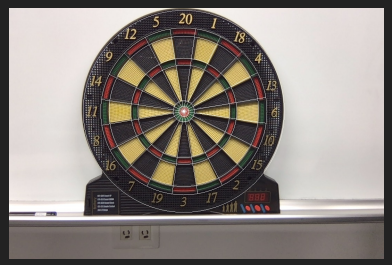
Project apex coordinate to Reference image

Landing coordinates on reference image (x, y)

Get score of that coordinate on Reference image

{ Attained score, 落下地点 (x, y) }

Input



Reference image

Image with thrown dart

Background image

output

Landing coordinate
(x, y)

Attained score