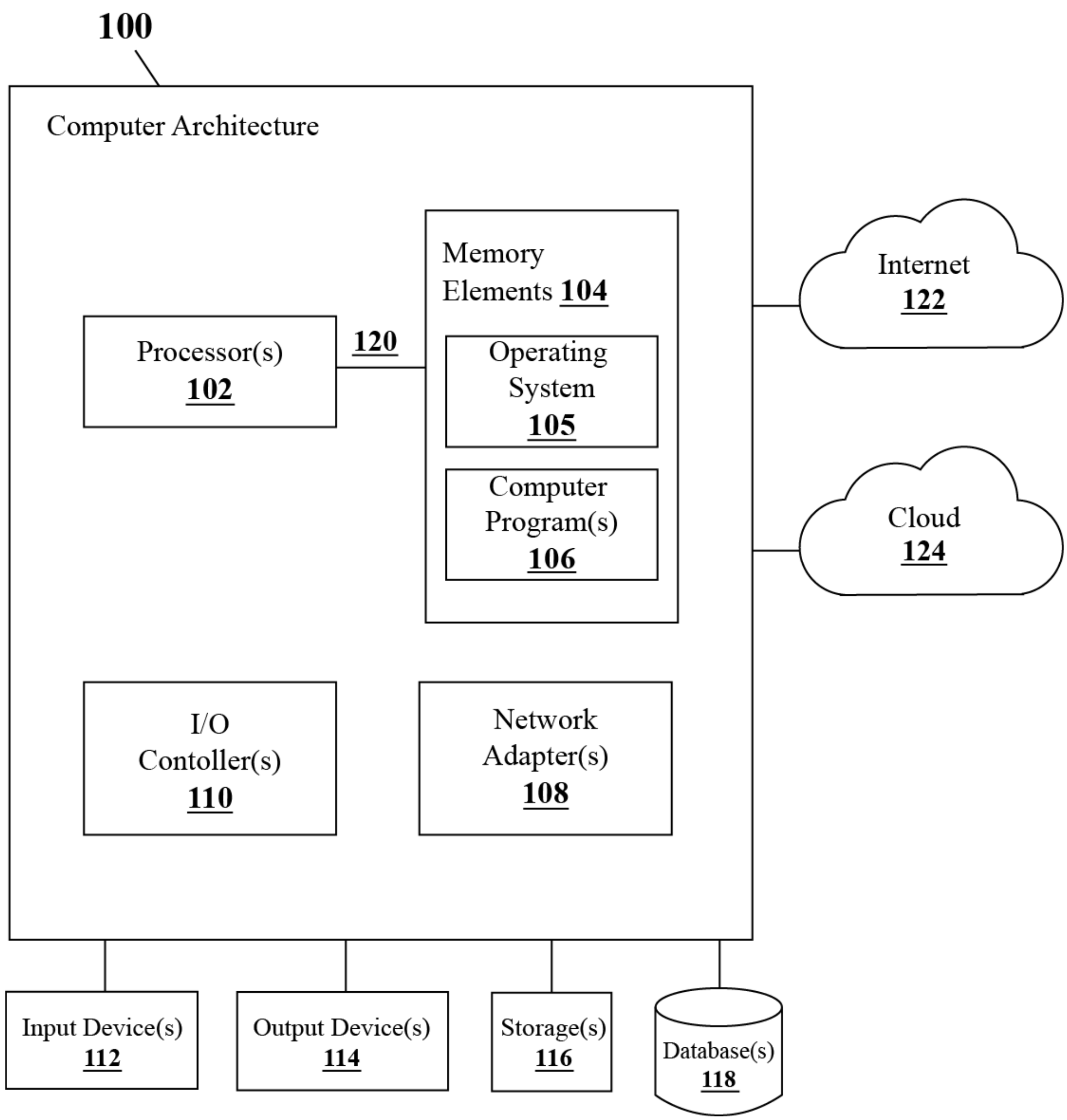
100
Computer Architecture
Memory
Elements 104
Processor(s)
102
120
Operating
System
105
Computer
Program(s)
106
I/O
Contoller(s)
110
Network
Adapter(s)
108
Internet
122
Cloud
124
Input Device(s)
112
Output Device(s)
114
Storage(s)
116
Database(s)
118

FIG. 1

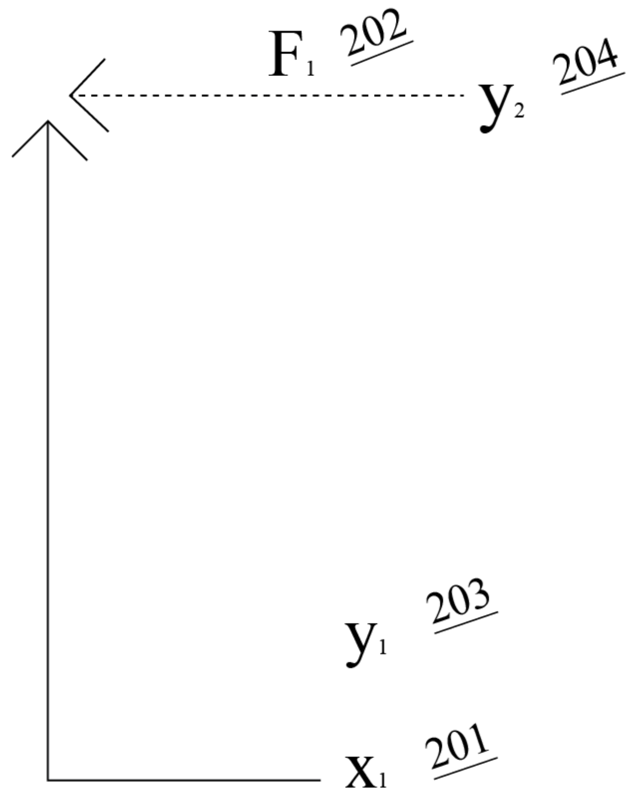

FIG. 2

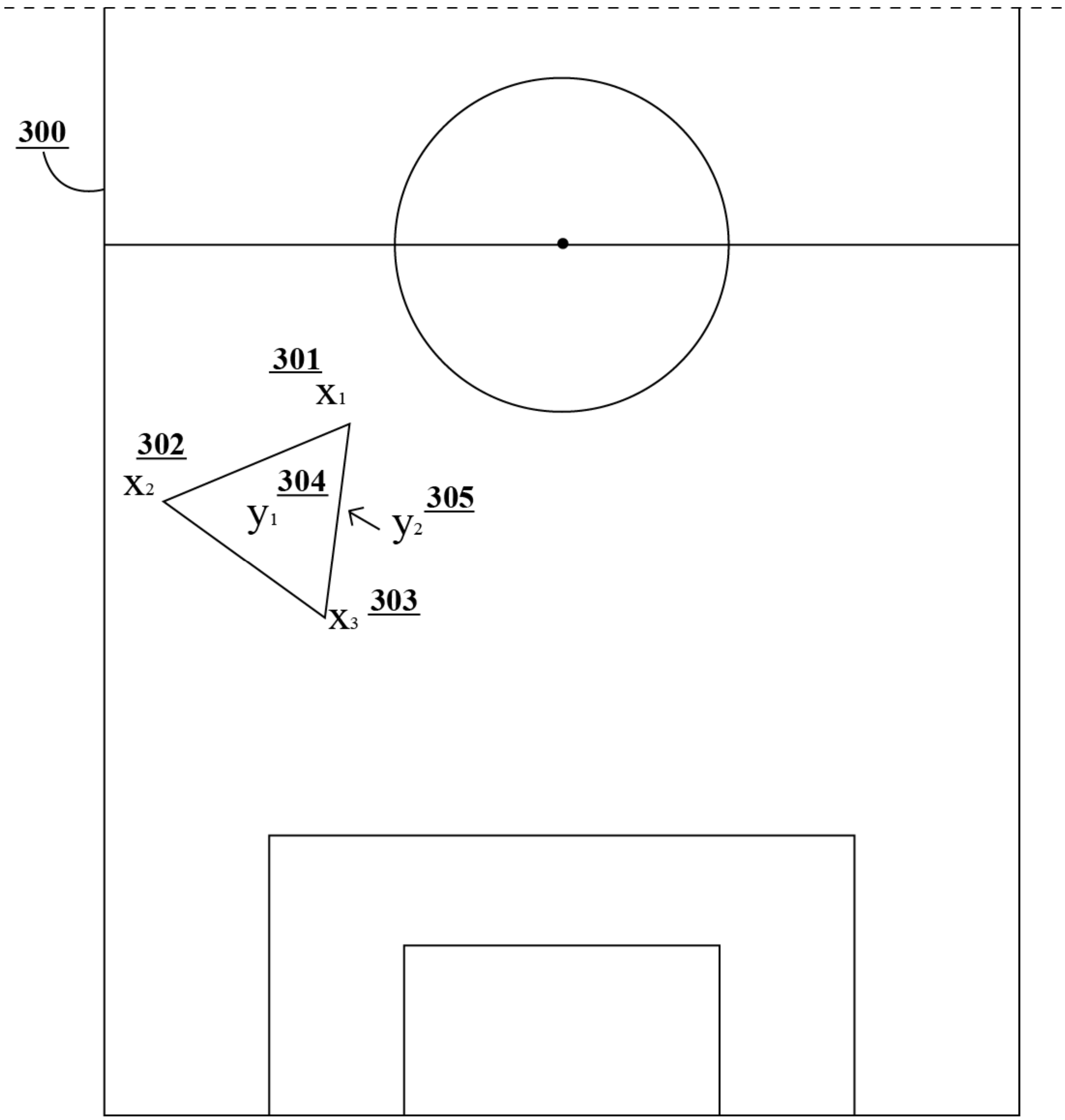

FIG. 3

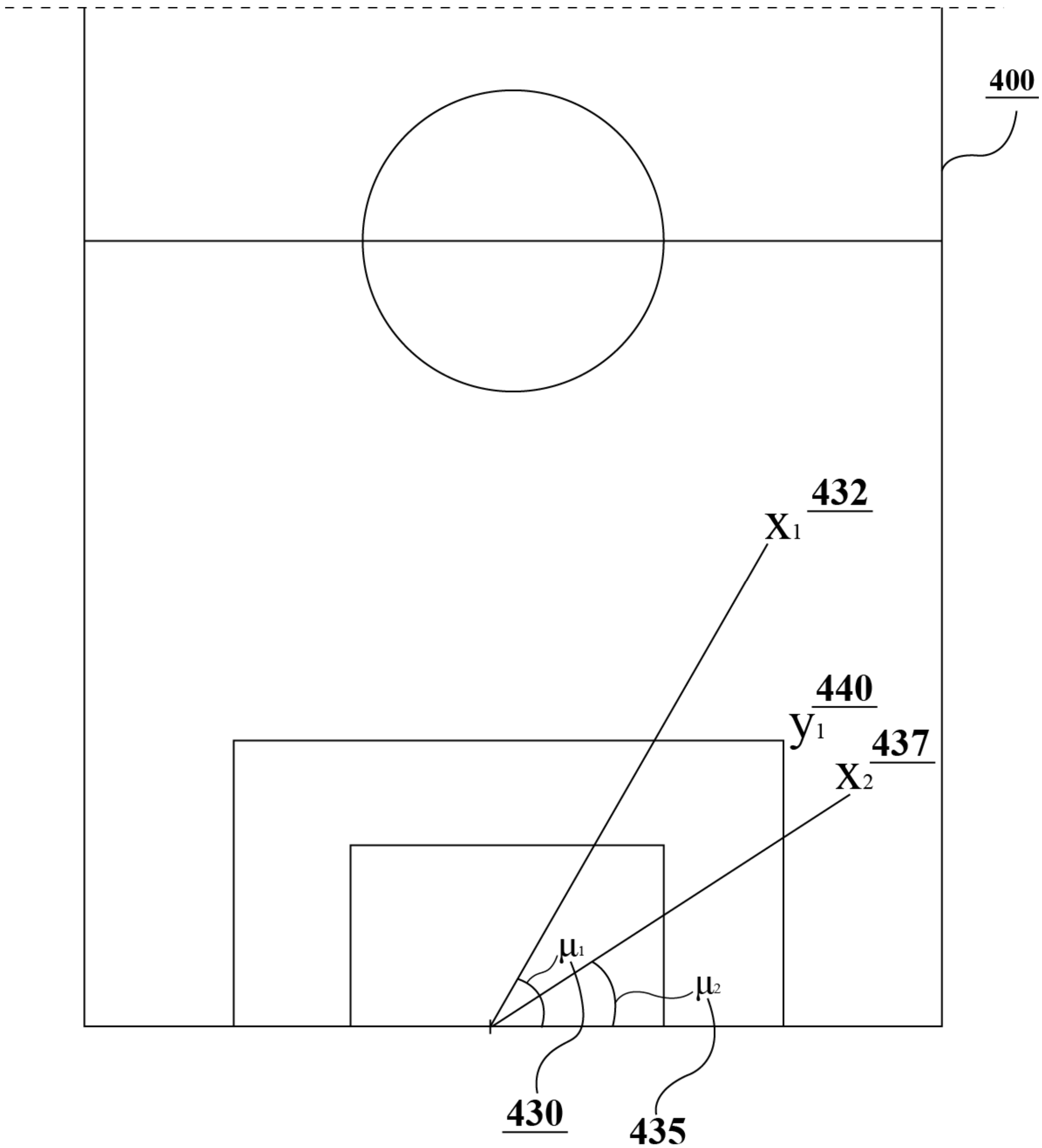

FIG. 4

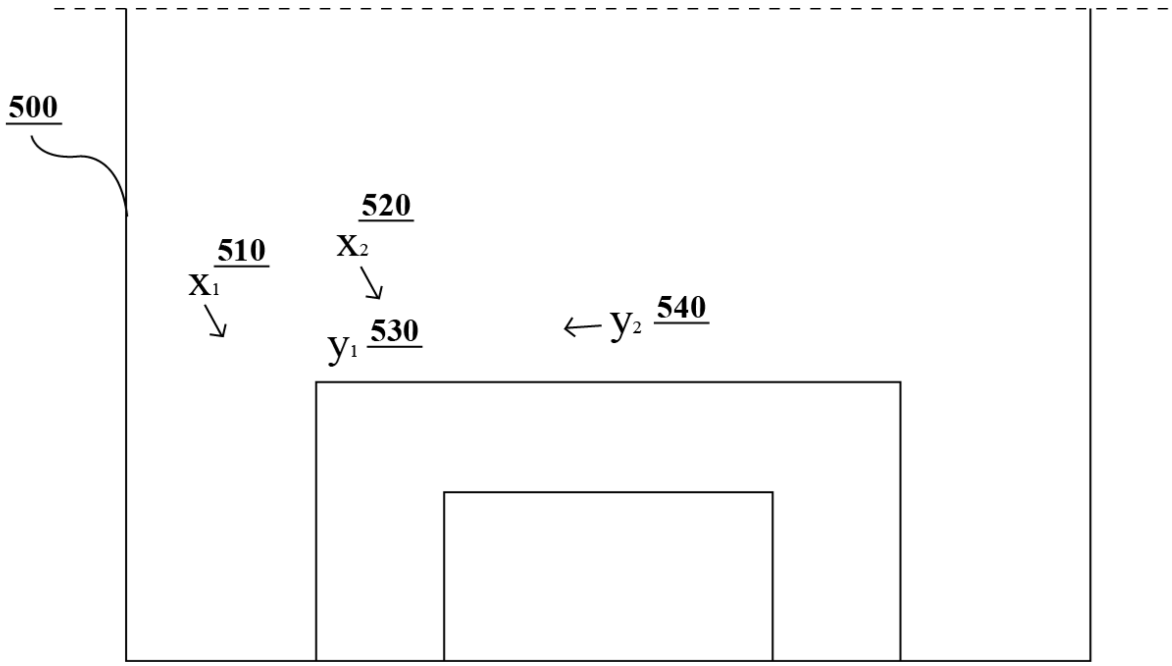

FIG. 5

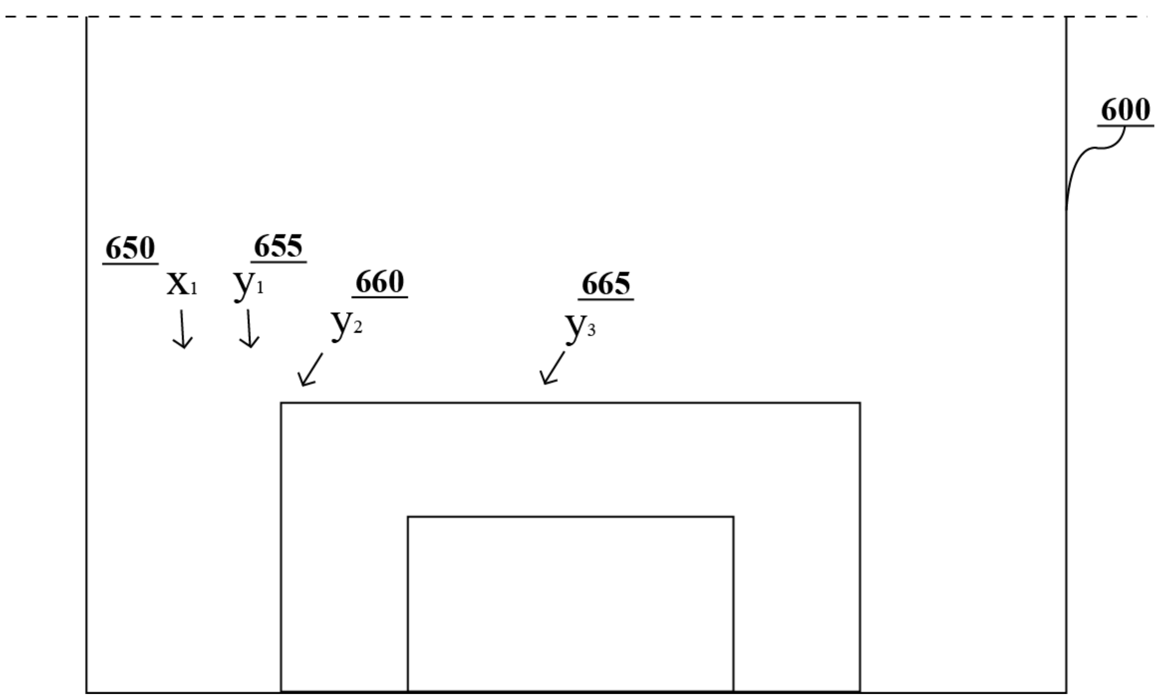

FIG. 6

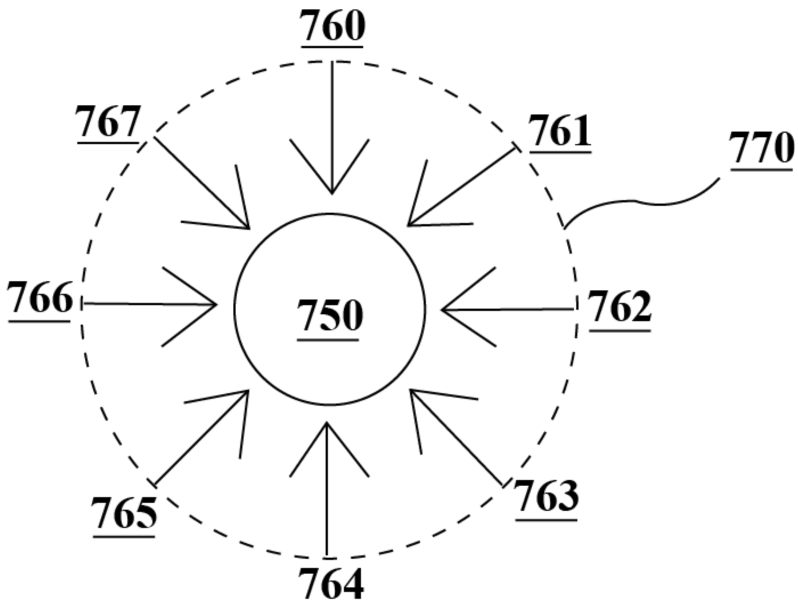

FIG. 7

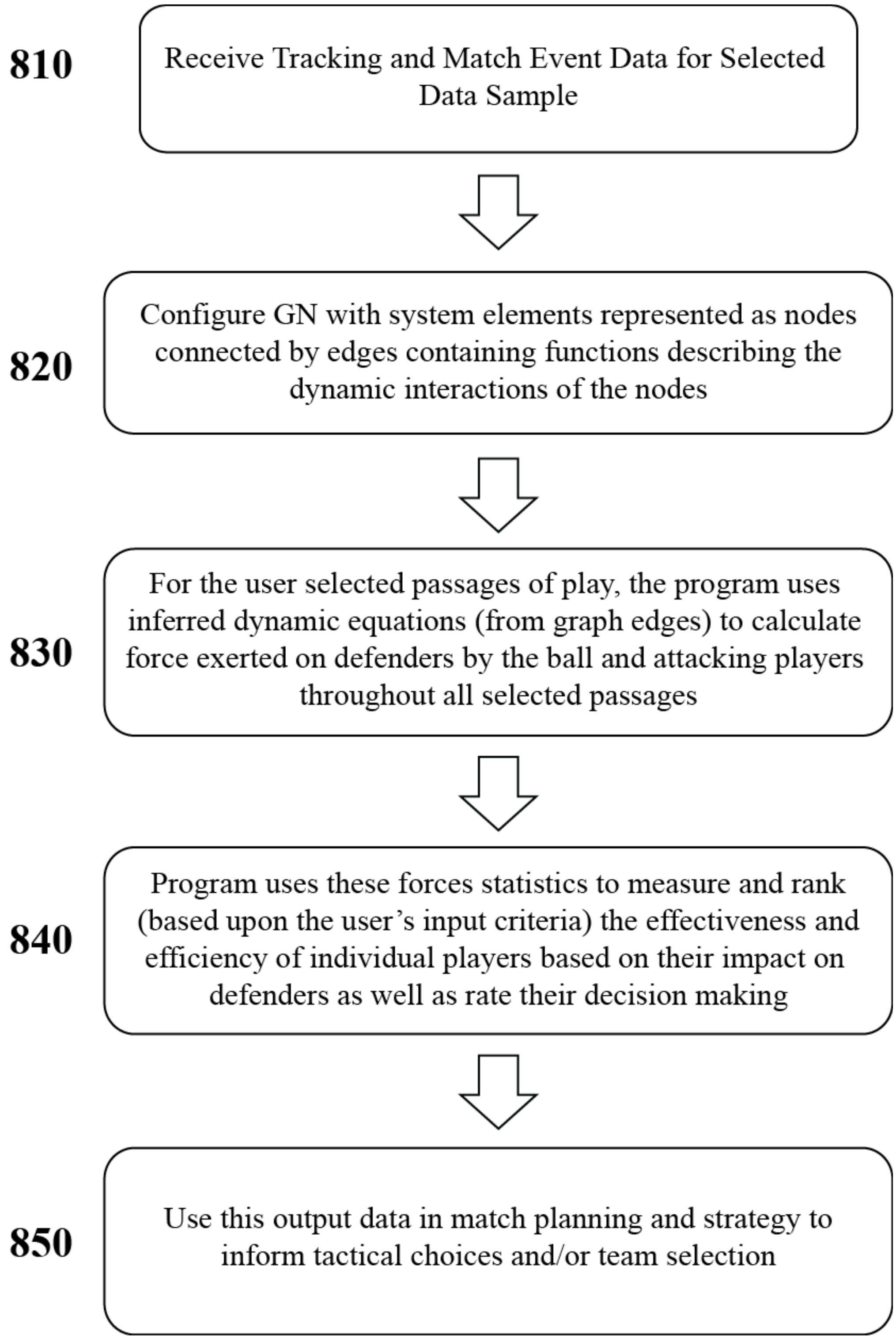
810
Receive Tracking and Match Event Data for Selected Data Sample
820
Configure GN with system elements represented as nodes connected by edges containing functions describing the dynamic interactions of the nodes
830
For the user selected passages of play, the program uses inferred dynamic equations (from graph edges) to calculate force exerted on defenders by the ball and attacking players throughout all selected passages
840
Program uses these forces statistics to measure and rank (based upon the user’s input criteria) the effectiveness and efficiency of individual players based on their impact on defenders as well as rate their decision making
850
Use this output data in match planning and strategy to inform tactical choices and/or team selection

FIG. 8

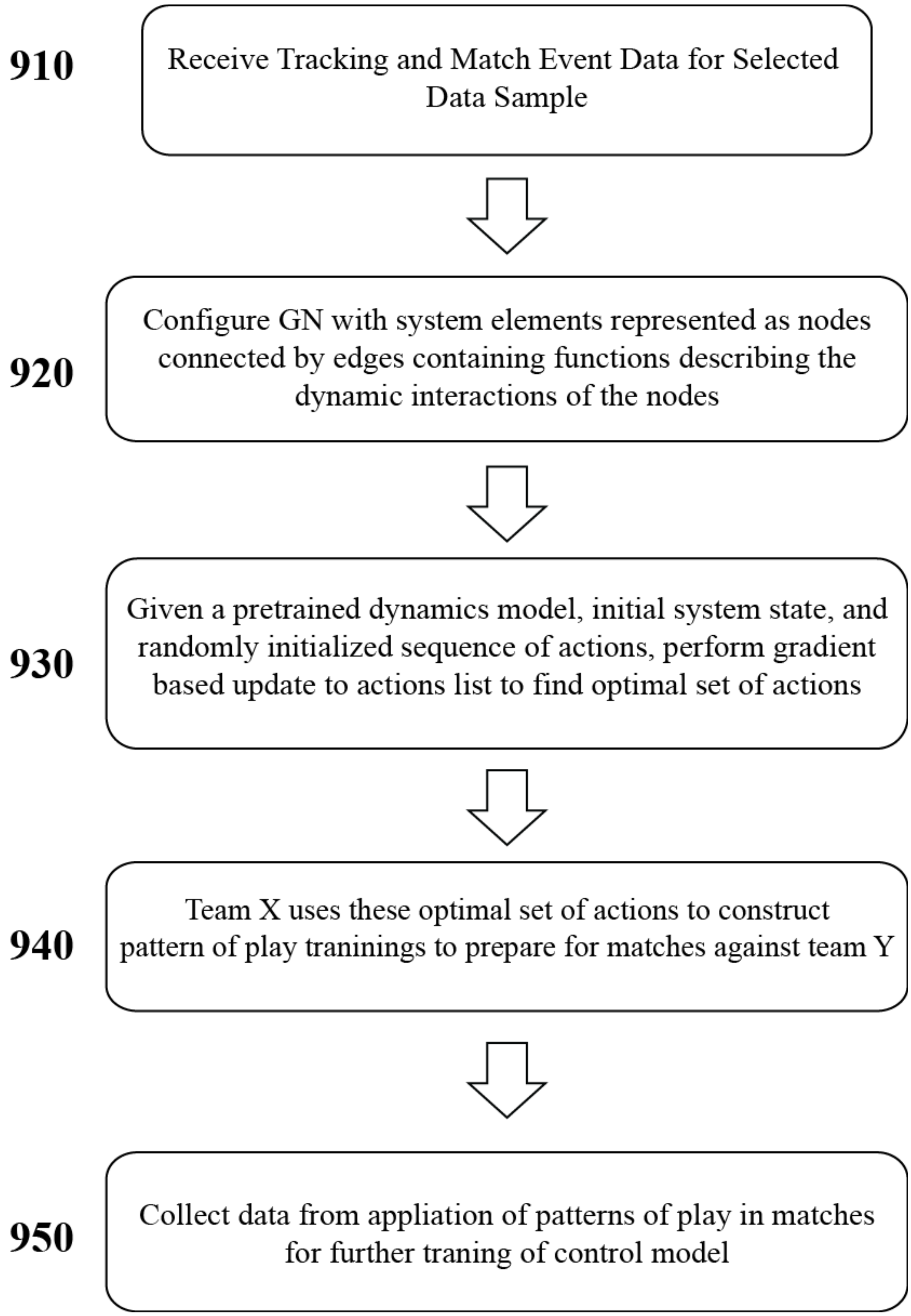
910
Receive Tracking and Match Event Data for Selected Data Sample
920
Configure GN with system elements represented as nodes connected by edges containing functions describing the dynamic interactions of the nodes
930
Given a pretrained dynamics model, initial system state, and randomly initialized sequence of actions, perform gradient based update to actions list to find optimal set of actions
940
Team X uses these optimal set of actions to construct pattern of play tranining to prepare for matches against team Y
950
Collect data from appliation of patterns of play in matches for further traning of control model

FIG. 9

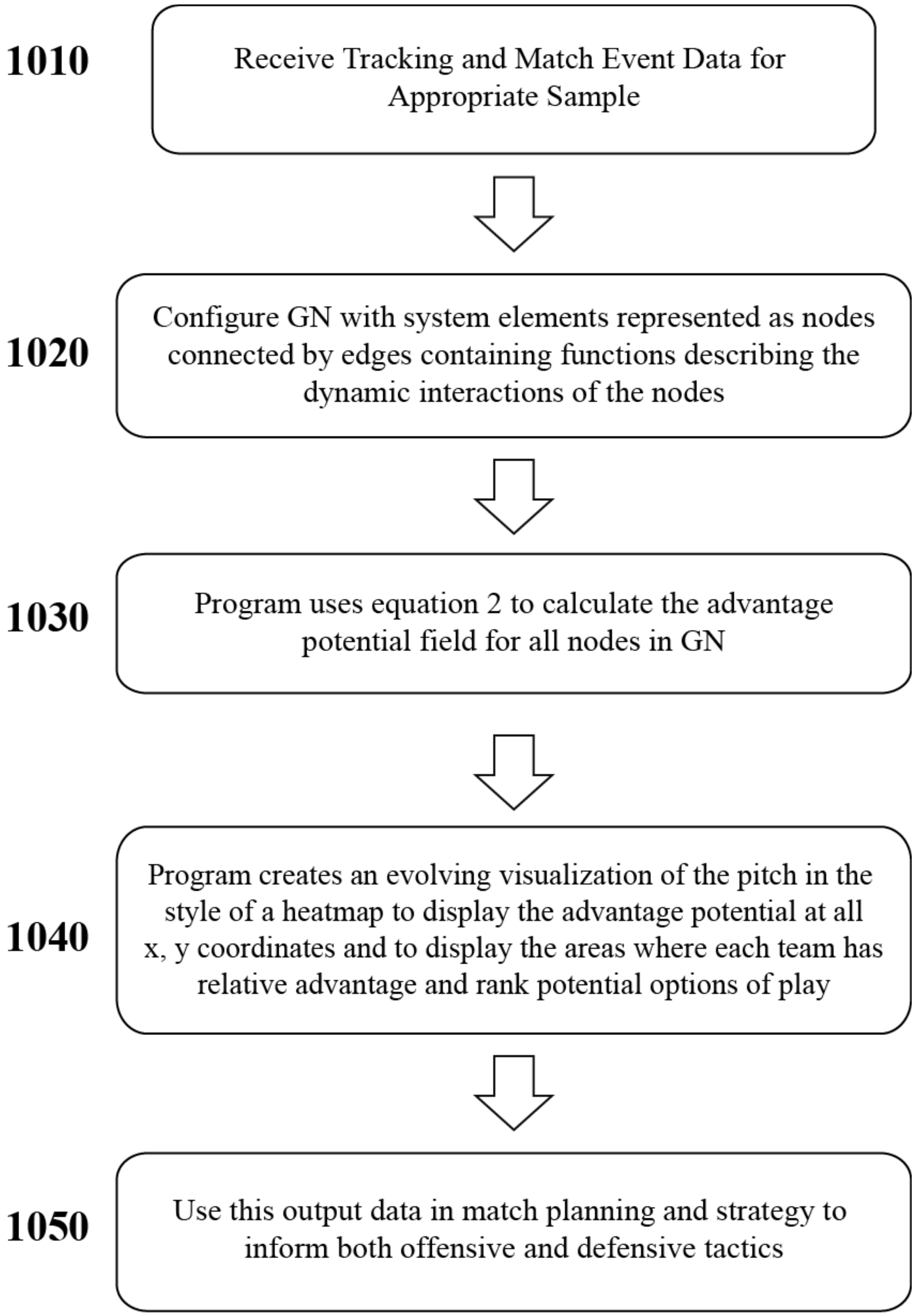
1010
Receive Tracking and Match Event Data for Appropriate Sample
1020
Configure GN with system elements represented as nodes connected by edges containing functions describing the dynamic interactions of the nodes
1030
Program uses equation 2 to calculate the advantage potential field for all nodes in GN
1040
Program creates an evolving visualization of the pitch in the style of a heatmap to display the advantage potential at all x, y coordinates and to display the areas where each team has relative advantage and rank potential options of play
1050
Use this output data in match planning and strategy to inform both offensive and defensive tactics

FIG. 10

Algorithm D.1 Forward prediction algorithm.

Input: trained GNs GN_1, GN_2 and normalizers Norm_{in}, Norm_{out}.
Input: dynamic state $\mathbf{x}^{t_0}$ and actions applied $\mathbf{x}^{t_0}$ to a system at the current timestep.
Input: system parameters $\mathbf{p}$
Build static graph G_s using $\mathbf{p}$
Build input dynamic nodes $N_d^{t_0}$ using $\mathbf{x}^{t_0}$
Build input dynamic edges $E_d^{t_0}$ using $\mathbf{a}^{t_0}$
Build input dynamic graph G_d using $N_d^{t_0}$ and $E_d^{t_0}$
Build input graph $G_i = \text{concat}(G_s, G_d)$
Obtain normalized input graph $G_i^n = \text{Norm}_{in}(G_i)$
Obtain graph after the first GN: $G^{'} = \text{GN}_1(G_i^n)$
Obtain normalized predicted delta dynamic graph: $G^* = \text{GN}_2(\text{concat}(G_i^n, G^{'}))$
Obtain normalized predicted delta dynamic nodes: $\Delta N_d^n = G^*.\text{nodes}$
Obtain predicted delta dynamic nodes: $\Delta N_d = \text{Norm}_{out}^{-1}(\Delta N_d^n)$
Obtain next dynamic nodes $N_d^{t_0+1}$ by updating $N_d^{t_0}$ with ΔN_d
Extract next dynamic state $\mathbf{x}^{t_0+1}$ from $N_d^{t_0+1}$
Output: next system state $\mathbf{x}^{t_0+1}$

Algorithm D.2 Forward prediction with System ID.

Input: trained parameter inference recurrent GN GN_p.
Input: trained GNs and normalizers from Algorithm D.1.
Input: dynamic state $\mathbf{x}^{t_0}$ and actions applied $\mathbf{x}^{t_0}$ to a parametrized system at the current timestep.
Input: a 20-step sequence of observed dynamic states x^{seq} and actions x^{seq} for same instance of the system.
Build dynamic graph sequence G_d^{seq} using x_i^{seq} and a_i^{seq}
Obtain empty graph hidden state G_h.
for each graph G_d^t in G_d^{seq} **do**
  $\text{G}_o, \text{G}_\text{h} = \text{GN}_p(\text{Norm}_{in}(G_d^t), \text{G}_\text{h})$,
end for
Assign $\text{G}_{ID} = \text{G}_o$
Use G_{ID} instead of G_s in Algorithm D.1 to obtain $\mathbf{x}^{t_0+1}$ from $\mathbf{x}^{t_0}$ and $\mathbf{x}^{t_0}$
Output: next system state $\mathbf{x}^{t_0+1}$

FIG. 11

Algorithm D.3 One step of the training algorithm

Before training: initialize weights of GNs GN_1, GN_2 and accumulators of normalizers Norm_{in}, Norm_{out}.
Input: batch of dynamic states of the system $\{\mathbf{x}^{t_0}\}$ and actions applied $\{\mathbf{a}^{t_0}\}$ at the current timestep
Input: batch of dynamic states of the system at the next timestep $\{\mathbf{x}^{t_0+1}\}$
Input: batch of system parameters $\{\mathbf{p}_i\}$
for each example in batch **do**
  Build static graph G_s using $\mathbf{p}_i$
  Build input dynamic nodes $N_d^{t_0}$ using $\mathbf{x}^{t_0}$
  Build input dynamic edges $E_d^{t_0}$ using $\mathbf{a}^{t_0}$
  Build output dynamic nodes $N_d^{t_0+1}$ using $\mathbf{x}^{t_0+1}$
  Add noise to input dynamic nodes $N_d^{t_0}$
  Build input dynamic graph G_d using $N_d^{t_0}$ and $E_d^{t_0}$
  Build input graph $G_i = \text{concat}(G_s, G_d)$
  Obtain target delta dynamic nodes $\Delta N'_d$ from $N_d^{t_0+1}$ and $N_d^{t_0}$
  Update Norm_{in} using G_i
  Update Norm_{out} using ΔN_d
  Obtain normalized input graph $G_i^n = \text{Norm}_{in}(G_i)$
  Obtain normalized target nodes: $\Delta N_d^{n'} = \text{Norm}_{out}(\Delta N'_d)$
  Obtain normalized predicted delta dynamic nodes: $\Delta N_d^n = \text{GN}_2(\text{concat}(G_i^n, \text{GN}_1(G_i^n))).\text{nodes}$
  Calculate dynamics prediction loss between ΔN_d^n and $\Delta N_d^{n'}$.
end for
Update weights of GN_1, GN_2 using Adam optimizer on the total loss with gradient clipping.

FIG. 12

Algorithm D.4 End-to-end training algorithm for System ID.

Before training: initialize weights of parameter inference recurrent GN GN_p, as well as weights from Algorithm D.3.
Input: a batch of 100-step sequences with dynamic states $\{x_i^{\text{seq}}\}$ and actions $\{x_i^{\text{seq}}\}$
for each sequence in batch **do**
  Pick a random 20-step subsequence x_i^{subseq} and a_i^{subseq}.
  Build dynamic graph sequence G_d^{subseq} using x_i^{subseq} and a_i^{subseq}
  Obtain empty graph hidden state $\mathrm{G_h}$.
  for each graph G_d^t in G_d^{subseq} **do**
    $\mathrm{G}_o, \mathrm{G_h} = \mathrm{GN}_p(\mathrm{Norm}_{in}(G_d^t), \mathrm{G_h}),$
  end for
  Assign $\mathrm{G}_{ID} = \mathrm{G}_o$
  Pick a different random timestep t_0 from $\{x_i^{\text{seq}}\}$, $\{x_i^{\text{seq}}\}$
  Apply Algorithm D.3 to timestep t_0 using final G_{ID} instead G_s to obtain the dynamics prediction loss.
end for
Update weights of GN_p, GN_1, GN_2 using Adam optimizer on the total loss with gradient clipping.

FIG. 13

Algorithm F.1 MPC algorithm

Input: initial system state $\mathbf{x}^0$,
Input: randomly initialized sequence of actions $\{\mathbf{a}^t\}$.
Input: pretrained dynamics model M such
$\mathbf{x}^{t_0+1} = M(\mathbf{x}^{t_0}, \mathbf{a}^{t_0})$
Input: Trajectory cost function L such
$c = C(\{\mathbf{x}^t\}, \{\mathbf{a}^t\})$
for a number of iterations **do**
  $\mathbf{x}_r^0 = \mathbf{x}^0$
  for t in range(0, horizon) **do**
    $\mathbf{x}_r^{t+1} = M(\mathbf{x}_r^t, \mathbf{a}^t)$
  end for
  Calculate trajectory cost $c = C(\{\mathbf{x}_r^t\}, \{\mathbf{a}^t\})$
  Calculate gradients $\{\mathbf{g}_a^t\} = \frac{\partial c}{\partial \{\mathbf{a}^t\}}$
  Apply gradient based update to $\{\mathbf{a}^t\}$
end for
Output: optimized action sequence $\{\mathbf{a}^t\}$

FIG. 14