Thesis overview

Ashesh Vasalya

May 20, 2019

—- paper – Predicting Object Transfer Position and Timing in Human-robot Handover Tasks

—-an idea –have velocity proportional to the distance between robot ef and subj w.r.t. handover location

Algorithm 1: linear prediction controller

Input: ${}^{ef}T_R, mocapData$

Output: ${}^{obj}wp_{ef}$ `// predicted waypoints`

Data: Initial require: $t_{observe} = 150ms, t_{predict} = 3sec, i = 1$

$i{+}{+}$ `// increments as per controller run-time (5ms)`

if $(i\%t_{observe}) == 0$ **then**

  ${}^{ef_{marker}}T_M \leftarrow mocapData.robotMarker((i - t_{observe}) + 1)$

  for $j \leftarrow 1$ *to* $t_{observe}$ **do**

    ${}^{obj_{marker}}T_M \leftarrow mocapData.objectMarker(i - t_{observe}) + j)$

    ${}^{obj_{marker}}T_{ef}(j) = {}^{ef}T_R^{-1} \times {}^{obj_{marker}}T_M \times {}^{ef_{marker}}T_M{}^{-1} \times {}^{ef}T_R$

    if $j == t_{observe}$ **then**

      ${}^{obj_{marker}}P_{ef}(t_{observe}) \leftarrow {}^{obj_{marker}}T_{ef}.translation()(t_{observe})$

  ${}^{h}\bar{\mathcal{V}}_{ef} \leftarrow \mathcal{F}_{avg}({}^{h}T_{ef}.translation())$

  `/* or */`

  ${}^{h}\bar{\mathcal{V}}_{ef} = \frac{1}{t_{observe}} \sum_{j=1}^{j=t_{observe}} ({}^{h}T_{ef}.translation())$

  `/* predict position of object handover at` $t_{predict}$ `*/`

  ${}^{h}P_{t_{predict}} \leftarrow {}^{h}\bar{\mathcal{V}}_{ef} \cdot t_{predict} + {}^{h}P_{ef}(t_{observe})$

  `/* generate way points between robot left end effector and object handover location */`

  Function `generateWp`$({}^{obj_{marker}}P_{ef}(t_{observe}), {}^{obj}P_{t_{predict}}, t_{predict})$**:**

    for $k \leftarrow 0$ *to* $t_{predict}$ **do**

      ${}^{obj}wp_{ef}(k) \leftarrow ({}^{obj_{marker}}P_{ef}(t_{observe}) - {}^{obj}P_{t_{predict}}) \times k - {}^{obj_{marker}}P_{ef}(t_{observe})$

    return ${}^{obj}wp_{ef}$

Algorithm 2: linear prediction controller - Position

Input: ${}^{ef}X_R, mocapData$

Output: ${}^{obj}wp_{ef}$ `// predicted waypoints`

Data: Initial require: $t_{observe} = 150ms, t_{predict} = 3sec, i = 1, dt = 5ms$

$i{+}{+}$ `// increments as per controller run-time (5ms)`

if $(i\%t_{observe}) == 0$ **then**

  for $j = 1$ *to* $t_{observe}$ **do**

    ${}^{h}X_M = mocapData.handMarker(i - t_{observe}) + j)$

    ${}^{h}X_{ef}(j) = {}^{h}X_M\,{}^{M}X_R\,{}^{ef}X_R{}^{-1}$

    if $j == t_{observe}$ **then**

      ${}^{h}\mathcal{P}_{ef}(t_{observe}) = {}^{h}X_{ef}.translation()(t_{observe})$

  ${}^{h}\bar{\mathcal{V}}_{ef} = \frac{1}{t_{observe}} \sum_{j=1}^{j=t_{observe}} f'({}^{h}\mathcal{P}_{ef})/dt$

  `/* predict subject's hand handover position at` $t_{predict}$ `*/`

  ${}^{h}\mathcal{P}_{t_{predict}} = {}^{h}\bar{\mathcal{V}}_{ef} \cdot t_{predict} + {}^{h}P_{ef}(t_{observe})$

  `/* generate way points between subject hand and robot left end-effector handover location */`

  Function `generateWp`$({}^{h}\mathcal{P}_{ef}(t_{observe}), {}^{h}\mathcal{P}_{t_{predict}}, t_{predict})$**:**

    for $k = 0$ *to* $t_{predict}$ **do**

      ${}^{h}wp_{ef}(k) = ({}^{h}\mathcal{P}_{ef}(t_{observe}) - {}^{h}\mathcal{P}_{t_{predict}}) \times k - {}^{h}\mathcal{P}_{ef}(t_{observe})$

    return ${}^{h}wp_{ef}$

Our prediction controller behavior can be tuned by two initially required constant time periods, $t_{observe}$ —which defines the time period required to observe the motion of object and $t_{predict}$ —required to predict the object handover location in advance.

Inputs of the controller are robot left end effector pose ${}^{ef}T_R$ and mocap markers position data in the mocap frame of reference $mocapFrameData$. ${}^{ef_{marker}}T_M$ is the left end effector marker pose in the mocap frame. Similarly, object marker pose given by ${}^{obj_{marker}}T_M$. For simplicity, at the moment, we have assumed zero rotation of the markers and end effector, therefore rotation part of ${}^{ef}T_R$, ${}^{efl_{marker}}T_M$ and ${}^{obj_{marker}}T_M$ are Identity matrix $\mathcal{I}$.

The transformation matrix ${}^{obj_{marker}}T_{ef}$, provides the relative pose of object marker w.r.t. robot left end effector in the robot coordinate system. $P_0^{obj_{marker}}$ $P_{t_{observe}}^{obj_{marker}}$ are the updated observed positions of object whenever the condition $(i\%t_{observe} == 0)$ satisfy. Based on the $P_0^{obj_{marker}}$ $P_{t_{observe}}^{obj_{marker}}$, average velocity $\bar{\mathcal{V}}_{obj_{marker}}^{ef}$ of the observed object motion calculated. Where, $\mathcal{F}_c$, $\mathcal{F}_{diff}$, $\mathcal{F}_{avg}$ are the *helper functions*.

Function $PREDICTPOS$, returns the predicted position of the handover $P_{t_{predict}}^{obj}$ at time $t_{predict}$.

Function $GENERATEWP$, returns the way-points wp_{obj}^{ef} between robot left end effector and object handover location, which is the final output of the controller.

Later, wp_{obj}^{ef} is being fed in the `mc_rtc positionTask`.

Algorithm 3: Force Based Gripper Controller

Input: $\mathcal{F}_w$ `// EF wrist worldWrenchWithoutGravity`

Output: $\mathcal{F}_{pull}, T_{new}$ `// Pull force, new threshold based on object mass`

$i{+}{+}$ `// increments as per controller run-time (5ms)`

if *subject hand is near robot* **then**

  `/* when SUBJECT holds the object */`

  if $\mathcal{F}_w.norm() < 1.0$ `// gripper is empty`

  then

    $\mathcal{F}_{zero} \leftarrow \mathcal{F}_w$ `// wrench offset`

  else if $\mathcal{F}_w.norm() > 2.0$ **then**

    $\mathcal{F}_{load} \leftarrow \mathcal{F}_w$ `// wrench with object`

  `/* when ROBOT holds the object */`

  Function `CheckPullForce(`$axis \forall x,y,z$`):`

    $objectMass \leftarrow \mathcal{F}_{load}/9.81$ `// get object mass`

    $\mathcal{F}_{inert} \leftarrow objectMass * efAce$ `//` $efAce$ `using gripper markers`

    $\mathcal{F}_{pull} \leftarrow |(|F_w - |F_{inert})$

    $T_{new} \leftarrow \mathcal{F}_{load} + T_{old}$ `//` T_{old} `set to min by user`

    if $\mathcal{F}_{pull} > T_{new}\ \forall x,y,z$ **then**

      openGripper `// release object`

else

  if $(i\%200)$ **then**

    $\mathcal{F}_{load} \leftarrow |(\mathcal{F}_w - \mathcal{F}_{zero})$ `// ∀x,y,z average over time`