HW2-Forward Kinematics

0856631 陳立軒

1 Introduction

Forward kinematics refers to the use of kinematics equations to compute the position of the end-effector form specified values for the joint parameters.

2 Fundamentals

Every bone in skeleton has its joint space, it can perform translation, rotation with respect to its local coordinate, and every system has only one global coordinate sharing the same global origin (0, 0, 0).

We get all motion information in AMC file, while the AMC file is specified with respect to the local coordinate (i.e. joint space), so all we need to do is converting it into global coordinate.

3 Implementation

3.1 Forward kinematics

In the implementation of FK, our goal is to convert joint space to global space, I first set the $start\_pos, end\_pos$ and $rotation$ of the root.

```
PoseColl_t pose_collection;
Pose root;
root.set_start_pos(joint_spatial_pos.at(0).linear_vector());
root.set_end_pos(joint_spatial_pos.at(0).linear_vector());
root.set_rotation(math::ComputeRotMatXyz(
    math::ToRadian(skeleton()->bone_ptr(0)->axis)
    )
);
pose_collection.push_back(root);
```

Second, I compute ${}_0^iR$ by chain rule ${}_0^iR = {}_0^1R \cdot {}_1^2R \cdots {}_{i-1}^iR$.

```
while (current_bone.idx != 0) {
    R_asf = math::ToRotMat(current_bone.rot_parent_current).transpose();
    R_amc = math::ComputeRotMatXyz(
        math::ToRadian(
            joint_spatial_pos.at(current_bone.idx).angular_vector()
        )
    );
    R_i_0 = R_asf * R_amc * R_i_0;

    current_bone = *current_bone.parent;
}
```

Finally, we can compute all bone's $start\_pos$ and end_pos by

$$_{i+1}T = {}_{0}^{i}RV_i + {}_{i}T$$

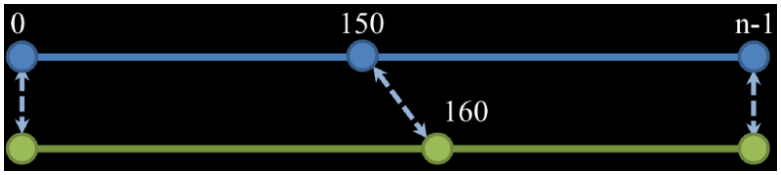
```
math::Vector3d_t V_i = skeleton()->bone_ptr(boneidx)->dir * skeleton()->bone_ptr(boneidx)->length;
math::Vector3d_t T_i = pose_collection.at(skeleton()->bone_ptr(boneidx)->parent->idx).end_pos();
math::Vector3d_t T_i_plus_1 = R_i_0 * V_i + T_i;

Pose temp;
temp.set_start_pos(T_i);
temp.set_end_pos(T_i_plus_1);
temp.set_rotation(math::ComputeRotMatXyz(
    math::ToRadian(skeleton()->bone_ptr(boneidx)->axis)
    )
);
```

In addition, the rotation information is stored in ASF file corresponded to the $axis$ field, which is represented as Euler angle with respect to the global coordinate, so we can just convert it into rotation matrix format and assign it.

3.2 Time warping

In the implementation of time warping, our goal is to apply hard constraint to let the character catch the ball and go back its origin position in the specific time period.



I first compute frame steps per second by

```
float frame_per_second = hard_constraint_coll_->at(2).frame_idx / hard_constraint_coll_->at(2).play_second;
```

and calculate the desired catching ball frame after warping.

```
int32_t warped_desired_catch_ball_frame_idx = frame_per_second * hard_constraint_coll_->at(1).play_second;
```

then I compute the frame steps before the character catching the ball.

```
float frame_step_before_catch_ball = (float)warped_desired_catch_ball_frame_idx / (float)desired_catch_ball_frame_idx;
```

Since the motion of catching ball is bring forward and the final position of the character is unchanged, so the frame steps after the character catching the ball must elongate.

```
float frame_step_after_catch_ball = (float)(hard_constraint_coll_->at(2).frame_idx - warped_desired_catch_ball_frame_idx) /
    (float)(hard_constraint_coll_->at(2).frame_idx - desired_catch_ball_frame_idx);
```

Since the frame id is probable not an integer, we don’t have motion information; so, we need to do interpolation.

```
float slerp_frame_idx = frame_step_before_catch_ball * frameidx;
int32_t prev_frame_idx = floorf(slerp_frame_idx);
int32_t next_frame_idx = ceilf(slerp_frame_idx);

math::Quaternion_t prev_Q = math::ComputeQuaternionXyz(
    math::ToRadian(original_motion_sequence_->element(boneidx, prev_frame_idx).angular_vector()).x(),
    math::ToRadian(original_motion_sequence_->element(boneidx, prev_frame_idx).angular_vector()).y(),
    math::ToRadian(original_motion_sequence_->element(boneidx, prev_frame_idx).angular_vector()).z()
);

math::Quaternion_t next_Q = math::ComputeQuaternionXyz(
    math::ToRadian(original_motion_sequence_->element(boneidx, next_frame_idx).angular_vector()).x(),
    math::ToRadian(original_motion_sequence_->element(boneidx, next_frame_idx).angular_vector()).y(),
    math::ToRadian(original_motion_sequence_->element(boneidx, next_frame_idx).angular_vector()).z()
);

double slerpRatio = slerp_frame_idx - prev_frame_idx;
math::Vector6d_t temp = original_motion_sequence_->element(boneidx, frameidx);
temp.set_angular_vector(math::ToDegree(
    math::ComputeEulerAngleXyz(
        math::ComputeRotMat(
            math::Slerp(prev_Q, next_Q, slerpRatio)
        )
    )
));
```

4 Conclusion

During the implementation of the forward kinematics and time warping, I learn the translation from joint space to global space, the capability of reading ASF and AMC files, the use of third-party API, and so on.