Contents

Presentation Script: The Robot Motion Kernel

3-Person Team Presentation (12-15 minutes)

Team Members: - **Jinsha Anna Antony** (Technical Lead) - **Katherine Rajala** (Implementation Lead)
- **Abhishek Kumar** (Results & Demo Lead)

Course: Robot Modelling
Institution: University of West Sweden, Trollhättan

Timing Breakdown

- **Jinsha**: 4-5 minutes (Slides 1-4)
- **Katherine**: 5-6 minutes (Slides 5-8)
- **Abhishek**: 3-4 minutes (Slides 9-11)
- **Total**: 12-15 minutes

JINSHA ANNA ANTONY - Introduction & Problem Analysis (4-5 min)

Slide 1: Title Slide

[Display title slide]

> "Good morning/afternoon everyone. I'm Jinsha, and together with Katherine and Abhishek, we'll present our project: **The Robot Motion Kernel**. This project is about converting ABB's industrial robot programming language, RAPID, into MATLAB to control the ABB IRB1600 robot."

[Pause, then continue]

> "We'll show you how we implemented mathematical foundations for robot motion, created two different movement strategies, and achieved impressive performance results."

Slide 2: Executive Summary

[**Point to key achievements**]

"Let me start with what we achieved. We had **100% implementation success** - all required functions working perfectly."

[**Point to each bullet**]

"We implemented three key functions: - **quat2rotMatrix** - converting quaternion rotations to matrices - **MoveL** - linear motion with smooth orientation changes - **MoveJ** - optimized joint-space motion

Our mathematical precision was excellent - rotation matrices with determinant exactly 1.0, and path accuracy with zero deviation."

[**Point to performance**]

"But most impressively, we achieved **15 times faster** repositioning with our MoveJ function, reducing execution time from 1.5 seconds down to just 100 milliseconds."

Slide 3: Problem Analysis - Initial State

[**Show Figure 1 - error message**]

"Here's where we started. This error message shows the main problem: the `quat2rotMatrix` function was completely undefined. The robot couldn't even start."

[**List the issues**]

"We identified six critical issues: 1. Missing quaternion conversion function 2. No MoveL implementation 3. Wrong file paths in the code 4. Incorrect frame references 5. No MoveJ function for efficiency 6. Missing smooth orientation interpolation"

[**Transition**]

"So we had to build everything from scratch. Let me explain our requirements."

Slide 4: Requirements & Strategy

[**Point to requirements section**]

"We organized our work into primary requirements - the must-haves - and secondary requirements."

"**Primary requirements:** - **R1**: Quaternion mathematics with perfect rotation properties - **R2**: Linear Cartesian motion with straight-line paths
- **R3**: Exact alignment with ABB RAPID behavior"

"**Secondary requirements:** - **R4**: Faster joint-space motion - **R5**: Comprehensive testing - **R6**: Professional documentation"

[**Point to strategy**]

"We used a four-phase approach: First mathematical foundations, then motion planning, followed by optimization, and finally documentation. Each phase built on the previous one."

[Transition to Katherine]

"Now I'll hand over to Katherine, who will explain how we implemented these solutions."

KATHERINE RAJALA - Implementation & Mathematics (5-6 min)

Slide 5: Mathematical Foundations - Quaternions

[Take over smoothly]

"Thank you, Jinsha. I'll walk you through how we implemented the mathematical core of our robot motion system."

[Show quaternion formula]

"Quaternions are a mathematical way to represent rotations using four numbers: w, x, y, and z. They avoid a problem called gimbal lock that happens with Euler angles."

[Point to the conversion formula]

"Our `quat2rotMatrix` function converts these quaternions into 3×3 rotation matrices. This matrix equation might look complex, but it's a standard mathematical transformation."

[Show Figure 2]

"Here's our visualization. The left side shows the rotation matrix as a color heatmap - you can see the structure clearly. The right shows the actual 3D axis rotation - notice how the red X-axis has rotated."

Slide 6: SLERP - Smooth Orientation

[Show SLERP visualization]

"When a robot moves, we need smooth orientation changes. That's where SLERP comes in - Spherical Linear Interpolation."

[Explain the concept simply]

"Think of it like this: instead of rotating the shortest way through 3D space - which would look jerky - SLERP rotates along the surface of a sphere, giving us smooth, natural motion."

[Point to Figure 5 - comparison]

"This figure compares three methods. The green line shows linear interpolation - it's mathematically incorrect. The blue line shows normalized linear interpolation - better but still approximate. Our red line shows true SLERP - perfect spherical motion. You can see our method maintains constant angular velocity."

Slide 7: MoveL - Linear Cartesian Motion

[Show MoveL visualization - Figure 4]

"Now let's see this in action. MoveL creates perfectly straight paths in Cartesian space."

[Walk through the algorithm]

"The algorithm is straightforward: 1. We interpolate positions linearly - that gives us the straight line 2. We use SLERP for smooth orientations 3. We solve inverse kinematics for each waypoint 4. The robot follows the path"

[Point to the figure]

"Look at this red trajectory - it's perfectly straight. Zero deviation over 41 centimeters. We use 30 waypoints, which means 30 inverse kinematics calculations. This takes about 1.5 seconds."

[Emphasize use cases]

"This is critical for tasks like welding, drawing, or cutting - anywhere you need a precise straight line."

Slide 8: MoveJ - Joint-Space Motion

[Show MoveJ visualization - Figure 6 and 7]

"But sometimes we don't need a straight line. Sometimes we just want to get from point A to point B quickly. That's MoveJ."

[Explain the difference]

"MoveJ works differently - it interpolates directly in joint space. We only calculate inverse kinematics twice: once for the start position and once for the end. Then we simply interpolate the joint angles."

[Point to Figure 7 - side-by-side comparison]

"This comparison shows the key trade-off. On the left, MoveJ's path is curved in Cartesian space - that's natural for joint interpolation. On the right, MoveL's path is perfectly straight."

[Highlight performance]

"But look at the performance difference: MoveJ only needs 2 IK calculations versus 30 for MoveL. That's why it's 15 times faster - just 100 milliseconds instead of 1500."

[Transition to Abhishek]

"Now Abhishek will show you our results and demonstrate the system working."

ABHISHEK KUMAR - Results, Testing & Demo (3-4 min)

Slide 9: Pentagon Drawing Demo

[**Take over - this is your highlight!**]

"Thanks Katherine. Now let me show you what we built in action!"

[**Show Figure 9 - pentagon path**]

"This is our robot drawing a perfect pentagon - a five-sided shape. The red path shows the complete trajectory. You can see all five sides are perfectly straight lines."

[**Keep it simple and enthusiastic**]

"This demonstrates everything working together: - Each side uses MoveL for straight-line drawing - Between sides, we use MoveJ for fast repositioning - The smooth orientation changes come from our SLERP implementation"

[**Point out the achievement**]

"The robot completed this entire sequence automatically - all five sides plus repositioning. This proves our RAPID-to-MATLAB conversion works perfectly!"

Slide 10: Test Results & Validation

[**Show Figure 3 - test results**]

"Of course, we needed to prove everything works correctly. Here are our test results."

[**Point to the green checkmarks - keep it simple**]

"All 18 tests passed - that's 100% success rate. Let me highlight the key ones: - **Rotation matrix tests**: Determinant is exactly 1.0 - mathematically perfect - **Orthogonality**: Error less than 10 to the power of negative 15 - that's essentially zero - **SLERP tests**: Smooth interpolation verified"

[**Don't get too technical - keep moving**]

"These numbers confirm our implementation is not just working, but working with scientific precision."

Slide 11: Performance & Conclusions

[**Show Figure 10 - performance chart**]

"Let me wrap up with our performance results."

[**Point to the bar chart**]

"This chart compares MoveL and MoveJ. The big difference is in IK calls: 30 versus just 2. That's why MoveJ is 15 times faster."

[**Summarize key conclusions - read these clearly**]

"**Our three main conclusions:**

First, we successfully converted ABB RAPID to MATLAB with 100% functionality. Everything works exactly as expected.

Second, we have two motion strategies: MoveL for precision and MoveJ for speed. Engineers can choose based on their needs.

Third, our mathematical foundation is solid - all matrices are orthonormal, all tests pass. This is production-ready code."

Slide 12: Final Summary

[**Bring it home - confident and clear**]

"To summarize our project:"

[**Read each point clearly**]

" **Complete implementation** - all three functions working perfectly

Mathematical precision - rotation matrices with perfect properties

15x performance improvement - from 1500 milliseconds to 100

10 professional figures - complete visual documentation

100% test success - every single test passing"

[**Final statement - look at audience**]

"This project demonstrates how industrial robot programming can be successfully translated into MATLAB, giving us flexibility for research and education while maintaining industrial-grade performance."

[**Smile**]

"Thank you! We're happy to answer any questions."

Q&A Preparation (All Team Members)

For Abhishek (Easy Questions You Can Handle):

Q: "How long did the pentagon drawing take?" > "The pentagon drawing took about 8 seconds total - 5 sides with MoveL for accuracy, plus repositioning with MoveJ for speed. It's a good demonstration of using both motion types together."

Q: "Can you explain what the red lines in the figures show?" > "Sure! The red lines show the tool center point trajectory - that's the path the robot's tip follows. For MoveL, you can see they're perfectly straight. For the pentagon, you see five straight sides connected together."

Q: "What tests did you run?" > "We ran 18 tests covering everything: rotation matrix validation, SLERP interpolation, and robot motion. All 18 passed, which gives us confidence the system works correctly."

Q: "Why is MoveJ so much faster?" > "MoveJ is faster because it only calculates inverse kinematics twice - at the start and end positions. MoveL needs to calculate it 30 times, once for each waypoint along the path. That's why MoveJ is 15 times faster."

For Jinsha (Complex/Mathematical Questions):

Q: "Why use quaternions instead of Euler angles?" Q: "How did you validate the mathematical correctness?" Q: "What are the computational complexity trade-offs?"

For Katherine (Implementation Details):

Q: "How did you implement SLERP?" Q: "What are the differences between your implementation and ABB RAPID?" Q: "How many waypoints did you use and why?"

Presentation Tips

For Abhishek Specifically:

1. **Slides 9-11 are highly visual** - let the pictures do most of the talking
2. **Use simple language** - "straight line" not "Cartesian trajectory"
3. **Point at the figures** - physically gesture to the screen
4. **Smile when presenting** - these are the success/results slides!
5. **If you forget something** - just look at the slide bullets and read them clearly
6. **For Q&A** - if you don't know, say "That's a great question - Jinsha/Katherine, would you like to answer that?"

For All Presenters:

1. **Practice transitions** between speakers
2. **Time yourselves** - stay within your time slots
3. **Make eye contact** with audience, not just the screen
4. **Speak slowly and clearly** - technical content needs extra clarity
5. **Use the laser pointer** or hand gestures to direct attention
6. **Pause after showing a new figure** - give audience time to look

---

Backup Slides (If Extra Time)

Technical Appendices

- MATLAB code snippets
- Detailed mathematical derivations
- Performance benchmarking data
- Repository information

If Questions About Future Work:

- Collision detection and avoidance
- Multi-robot coordination

- Real-time trajectory optimization
- ROS integration for real hardware

Emergency Shortcuts (If Running Over Time)

Jinsha can skip: Detailed requirements list (just say "we had clear requirements")
Katherine can skip: SLERP comparison details (just show figure, say "SLERP gives smooth rotation")
Abhishek can skip: Detailed test numbers (just say "all 18 tests passed")

Good Luck!

Remember: - You did excellent work - The results speak for themselves - The figures are professional and clear - 100% success rate is impressive - Stay confident and have fun!