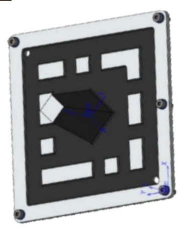
AI-driven **pose estimation** for **spacecraft** in-orbit servicing using FPGA acceleration

Silvia Romero Azpitarte[1,2], Javier Ferre Martín, Agustín Fernández León[3]
[1] Universidad Carlos III de Madrid,
[2] GMV Aerospace and Defence S.A.U.,
[3] European Space Agency (ESA), ESTEC

Introduction

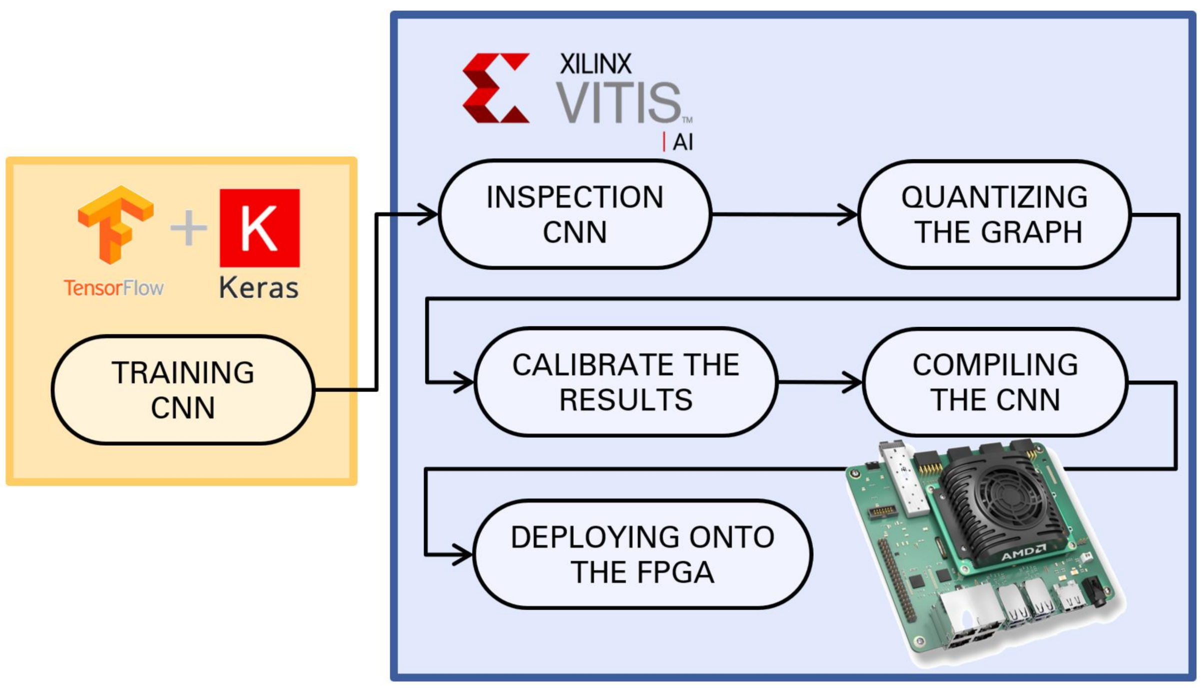
GMV's in-orbit servicing project requires an **enhancement in the classic pose estimation method** between two approaching vehicles with these features. In this poster the proposed **DL method for the close-range** phase of the rendezvous is presented.

Monocular camera



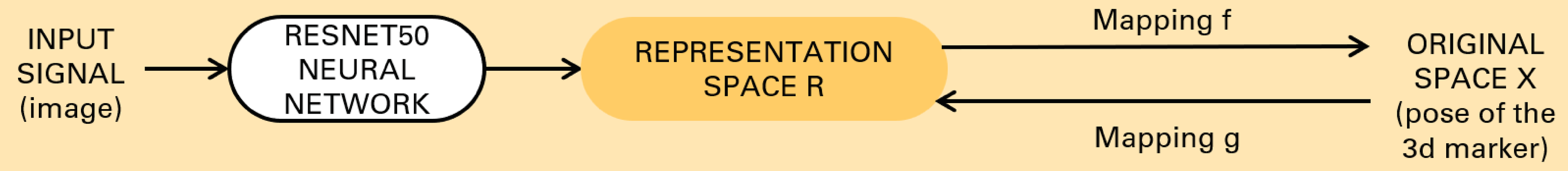
3D fiducial markers

Development



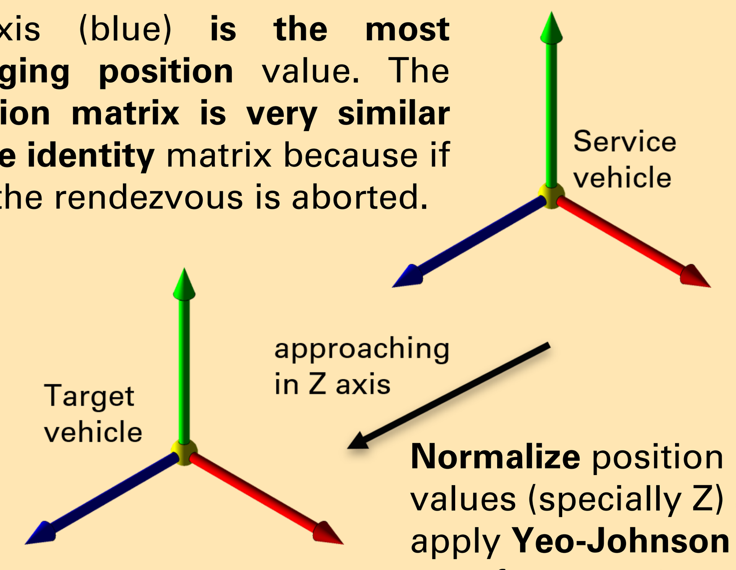
Conclusion

Accuracy in **nominal situations equals classic approach.** However, pose estimation in conditions with poor lighting and shadows, sudden illumination changes and partial occlusions is enhanced.
Therefore, this approach shows an increase in **robustness**. There is a risk of **overfitting** which doesn't apply to classic pose estimation.



Rotation, which is the most **difficult** learnable feature, is presented as the **first two rows of the DCM** (direction cosine matrix), since the third row is linearly dependent.

Z axis (blue) **is the most changing position** value. The **rotation matrix is very similar to the identity** matrix because if not, the rendezvous is aborted.



$$[\mathrm{pos_x}, \mathrm{pos_y}, \mathrm{pos_z}, R1_1, R1_2, R1_3, R2_1, R2_2, R2_3]$$

Gram-Schmidt process

$$P = [\mathrm{pos_x}\ \mathrm{pos_y}\ \mathrm{pos_z}]$$

$$R = \begin{bmatrix} R1_1 & R1_2 & R1_3 \\ R2_1 & R2_2 & R2_3 \\ R3_1 & R3_2 & R3_3 \end{bmatrix}$$

Normalize position values (specially Z) and apply **Yeo-Johnson** transform to $R1_1$ and $R2_2$

Train	Valid.	Test
60%	20%	20%
49,369	16,456	16,456

Out of the **82k+ images from the 52 trajectories in the CAT dataset** of two approaching satellites, 60% were used for train., 20% for val., and 20% for test. to help **prevent overfitting**.

Loss function has two **learnable** parameters $\widehat{\sigma_r}$ and $\widehat{\sigma_t}$ to **balance** learning between translation and rotation:

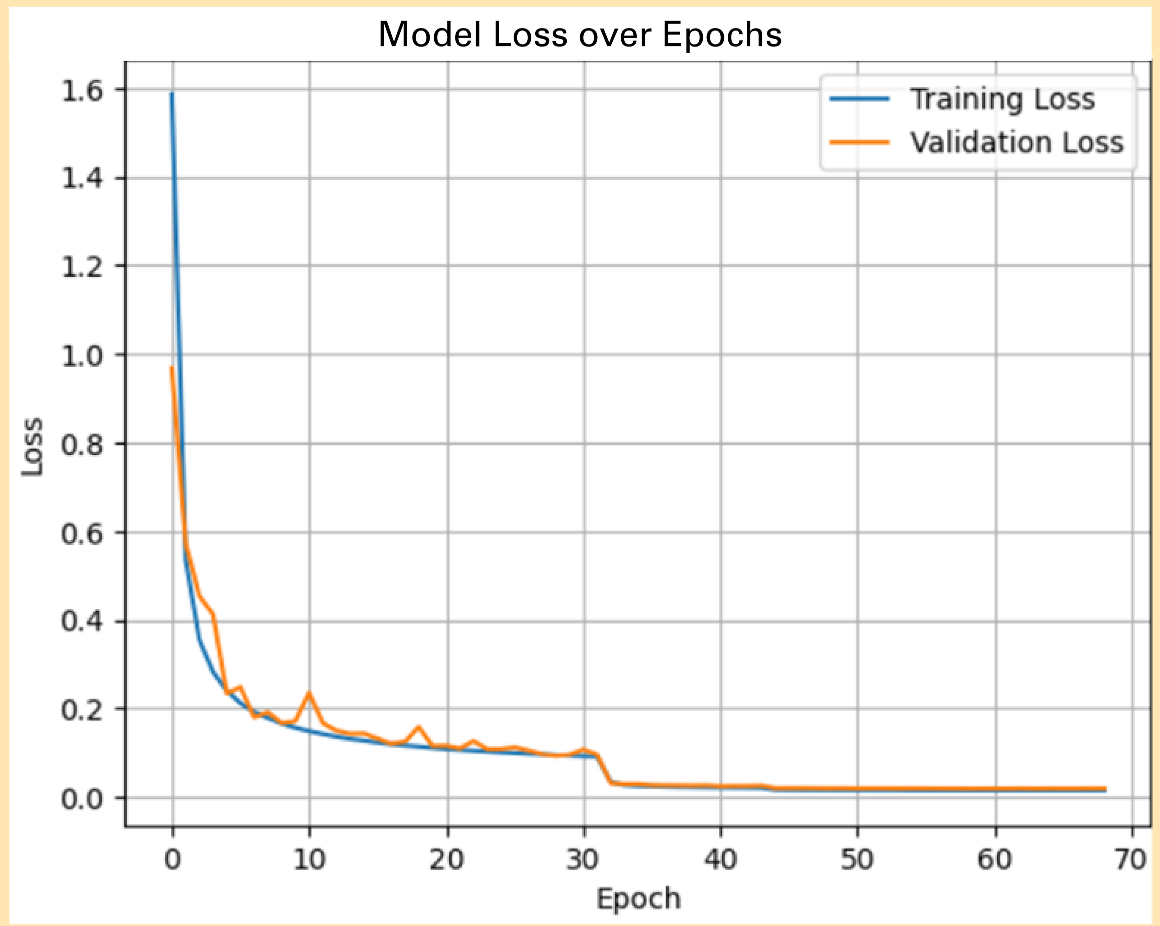
$$\mathcal{L} = L_r \exp(-2\widehat{\sigma_r}) + L_t \exp(-2\widehat{\sigma_t}) + 2(\widehat{\sigma_r} + \widehat{\sigma_t})$$

where $L_r = \sum_{i=1}^{batch} \|\widehat{r_i} - r_i\|$ and $L_t = \sum_{i=1}^{batch} \|\widehat{t_i} - t_i\|$

and || || denotes L2 norm. $\widehat{r_i}$ denotes predicted rotation and r_i denotes ground truth rotation. The same applies for translation.

Error function is:

- **L2 norm** between P_{true} and $P_{predicted}$ for position
- **Angle** between R_{true} and $R_{predicted}$ for orientation



final train loss: 0.0154 - final validation loss: 0.0187

AI model was deployed on the **Kria KR260**, leveraging the Zynq UltraScale+ MPSoC and **DPU** (hier_dpu) for hardware acceleration (target DPU is DPUCZDX8G_ISA1_B512). A custom PetaLinux environment was created to include DPU and ensure efficient deployment. The implementation was developed using Vivado 2022.2 and Vitis AI 3.5.

DPU	C	ZD	X	8	G	ISA1	B512
DPU IP core	CNN (ResNet50)	Hardware platform is Zynq DDR	Quantization method is DECENT™ (Deep Compression Tool)	Quantization bitwidth is 8-bit	General purpose	Instruction Set Architecture version	512 DSP blocks allocated

Resource	Utilization	Percentage in KRIA
LUT	30030	25.64%
LUTRAM	3602	6.25%
FF	41171	17.58%
URAM	18	28.13%
DSP	134	10.74%
BUFG	4	1.14%
PLL	1	12.50%

Platform	Latency	Nº instances evaluated	Avg time per instance
GPU*	126,000 ms	16,456	7.657 ms
FPGA	30,100 ms	100	301 ms

Future work includes a **reduction of the inference** time given the resource utilization.

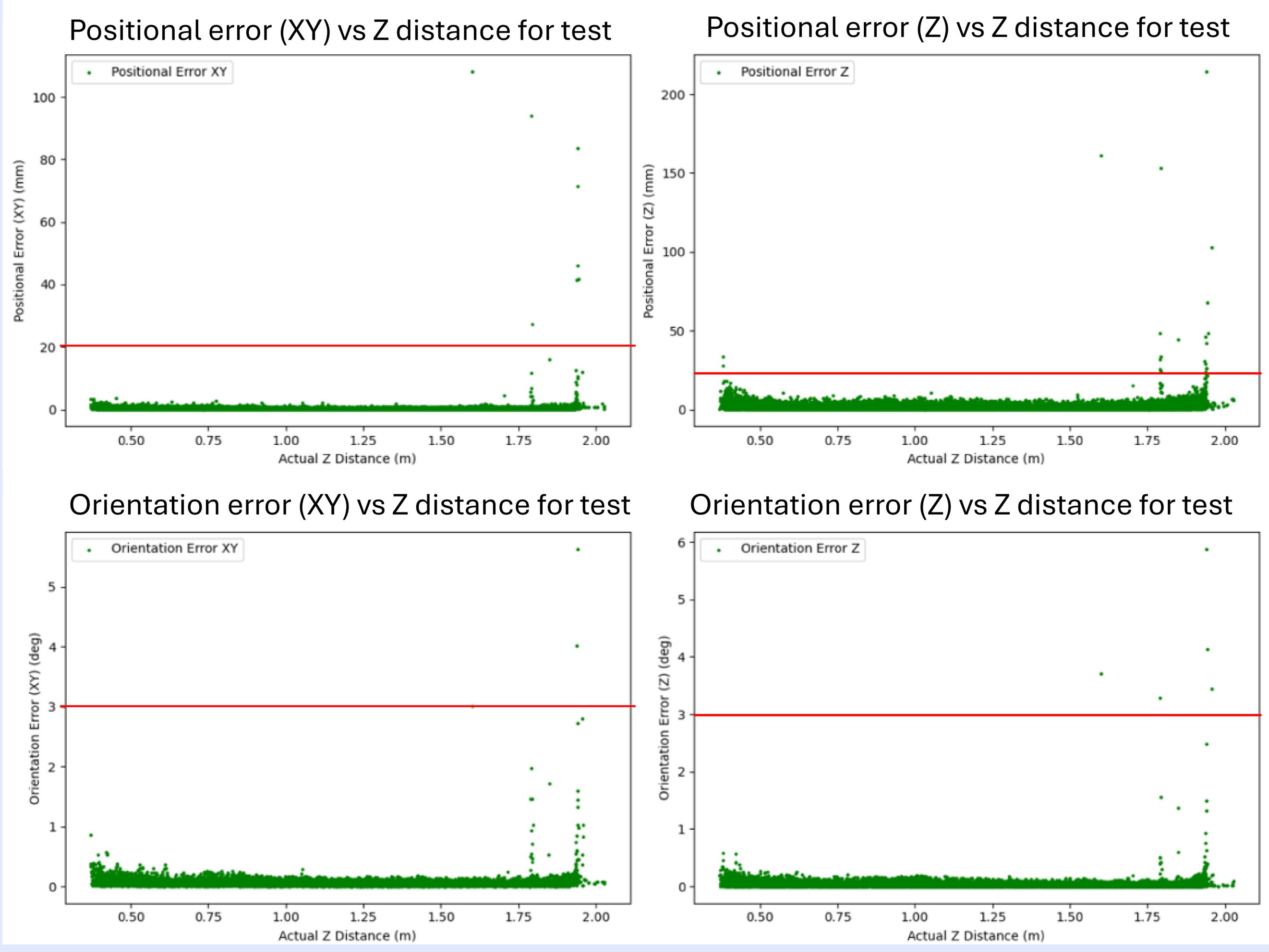
State	Avg Power Consumption
KRIA KR260 Idle	3.820 W
KRIA KR260 Vivado post-implementation prediction	3.886 W
KR260 Running model	4.030 W
GPU* Running model	73.550 W

Quantization transforms weights representation from **float32 to uint8**. Batch Normalization **layers are also folded** into the previous Convolutional layers. The model is then calibrated using the validation dataset and the loss function. The **loss is 0.3406 for the quantized model** over val. data, whereas for the original model the loss was 0.0187 for val. data.

The **error function** then calculates the deviation between ground truth and prediction for test data. This function shows a **performance degradation of 4.22% for position and 10.96% for orientation** of the quantized model on the FPGA compared to the original model on the GPU.

After quantization, fewer than 30 of 16,456 test images failed, mainly in extreme illumination cases. The model **remains robust** despite quantization and hardware transition. The model outperformed the classic CAT method in handling occlusions, lighting changes, and sudden shifts. **Future work could merge both methods or apply filtering** for added reliability.

Error Function Results & Allowed Errors by Axis vs. Actual Z Distance



Error	Total avg. absolute error		Allow. Error (red lines)
	GPU*	FPGA	
Position	2.011 mm	2.096 mm	20 mm
Rotation	0.073 deg	0.081 deg	3 deg

*NVIDIA GeForce RTX 3070 (5888 CUDA cores). 8GB memory. Driver version 560.35.03. CUDA version 12.6

Future **migration to a space-certified** platform could be achieved using **either a NanoXplore FPGA**, which would require a complete redesign due to its unique architecture and lack of compatibility with the Vitis AI workflow, **or a radiation-tolerant FPGA from AMD Xilinx**. The latter includes options such as the XQR Versal, which offers built-in AI acceleration, improved fault tolerance, and a more familiar development environment, potentially allowing for partial reuse of the existing workflow.