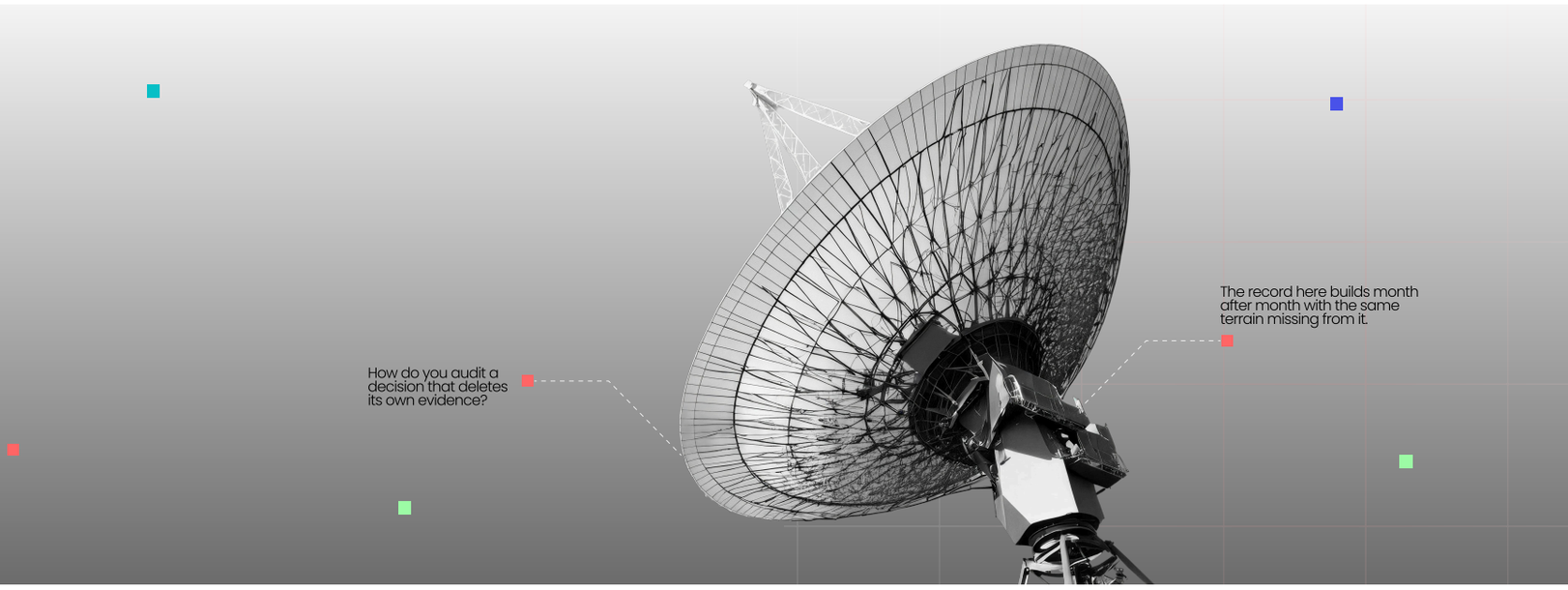
Case Study - Aerospace

HOW AN ONBOARD CLOUD FILTER DECIDES WHICH SATELLITE IMAGES TO KEEP

An aerospace illustration of where that decision is reliable, where it is not, and how Squint Vision Studio tells the two apart.



A growing number of Earth observation satellites now choose for themselves which of their own pictures are worth sending home.

Bandwidth is what drives this. These satellites capture far more imagery than they can transmit. Downlink windows are short, and a large share of every orbit passes over cloud. A thick cloud deck makes an optical image useless for most purposes, so the satellite runs a neural network onboard, detects cloud cover, and drops the heavily clouded scenes before they reach the downlink queue. The savings are large enough that onboard image selection has moved from experiment to standard practice.

This case study looks at the edge of that decision. It is about the scenes the model gets wrong, and about a property that makes those errors unlike most model failures. Nobody on the ground ever sees the mistake, because the evidence was deleted in orbit.

AN IRREVERSIBLE DECISION

A wrong prediction on the ground can usually be caught. The output gets logged, someone can review it, and a bad call can be revisited when new evidence turns up.

An onboard discard has none of that. When the model marks a scene as too cloudy to keep, the scene is dropped rather than flagged, and the bandwidth it would have used goes to the next image. Avoiding that transmission was the entire point, so no copy waits on the ground. The decision is autonomous, immediate and final in the strongest sense, because the data stops existing.

That changes what a misclassification costs. In most inspection work, a false alarm costs a re-check and a missed defect leads to a downstream failure. Here a wrong discard destroys something unrecoverable. It was a specific image of a specific place at a specific moment, from a satellite that will not pass overhead again for days, and it may have been the only chance to capture an event while it was happening.

WHERE THE MODEL FAILS

Cloud detection looks straightforward until you examine the cases sitting between obvious cloud and obvious ground. Those are where the consequences gather, and they come in two kinds that call for different handling.

The first kind is ambiguity, where more than one reading genuinely fits the evidence. Snow and ice are the classic trap. Snow-covered terrain and high cirrus cloud are both bright and cold, and in the spectral bands a small satellite carries they can look nearly identical. Snow and cloud confusion comes up repeatedly in the cloud detection literature as a primary source of error. Bright desert surfaces cause false positives for much the same reason. Thin cirrus is the opposite problem, being wispy and semi-transparent with low contrast, easy to miss altogether and easy to mistake for haze over clear ground. In all of these the data supports two answers and offers no way to choose between them.

The second kind is novelty, where the model has no adequate representation of what it is looking at. A surface type absent from training, an unusual atmospheric condition, a sensor artefact the model has never encountered. Here the problem is not competing interpretations but the absence of any grounded one.

A model trained mostly on unambiguous scenes learns those scenes well and reports high accuracy. Solid cloud over dark ocean and clear sky over green land are both easy. The harder cases are underrepresented in training, which is where the model's reasoning runs thin.

What matters is which way the error points. A model that has learned to associate bright and textured with cloud will look at a snow-covered mountain range under clear skies and confidently call it cloud. That scene might be exactly what a glaciologist or a disaster response team was waiting for. The model discards a clear, high-value image because the surface resembled the thing it was trained to throw away.

It does this with high confidence, because confidence measures how strongly the model prefers the cloud label rather than whether it understood the scene. A confident correct discard of a useless cloudy frame and a confident wrong discard of a rare clear one look the same coming out. In this setting, that output is the last anyone will know of either.

The failure also stays hidden after the fact. Operators see the images that arrive and never see the ones that were discarded, because those were deleted specifically so they would not occupy a downlink slot. The system produces no error signal and nothing unusual in its telemetry, because it is behaving exactly as designed, and that design leaves no trace of what it rejected. A slow bias against one kind of legitimate surface would show up as a quiet, permanent gap in the data record that nobody on the ground is positioned to notice.

SQUINT VISION STUDIO

An autonomous system with no human in the loop is making irreversible decisions with no way to know which of them sit in territory where its judgment fails. Squint Vision Studio works out where that territory is.

Before launch, Vision Studio modifies the model so it outputs its internal representations, called embeddings, alongside its predictions. Those embeddings are mapped into a view of the model's decision space, which separates the regions where its cloud and clear reasoning is well supported from the regions where inputs are sparse, overlapping or unfamiliar and reliability falls away.

Analysis of that space is what shows which conditions land where, rather than assumptions made in advance about which scene types will prove difficult. On the evidence from cloud detection research, the snow, ice, bright desert and thin cirrus cases are the ones you would expect to surface as problematic, and the map either confirms that or corrects it.

That work happens on the ground, ahead of the mission. It turns a vague worry about wrongly discarded images into a specific account of the surface and atmospheric conditions the model cannot be trusted on.

Vision Studio then exports a watchdog, a lightweight module that runs alongside the detection model rather than replacing it. When a scene about to be discarded lands outside the trusted regions, the watchdog flags it as a call the model is not equipped to make reliably, and the mission system decides what follows. The scene can be preserved and downlinked despite the cloud verdict, tagged for priority transmission, or held rather than deleted.

The two kinds of failure warrant different treatment. An ambiguous scene, where snow and cirrus both fit, is a case for keeping the data so the question can be settled on the ground with better tools than the satellite carries. A genuinely novel scene, where the model has no basis for any answer, is a case for preserving the image and marking it for review, since it may indicate a gap in the training distribution worth closing. Declining to act under uncertainty is a deliberate behaviour here rather than a workaround, and Squint's own framework treats abstention as a legitimate designed response.

Obvious cloudy frames are still discarded, which preserves the bandwidth savings that justified onboard autonomy in the first place.

WATCHING THE PATTERN OVER TIME

A single flagged scene is a small event. The more serious failure builds slowly, as a persistent bias against one kind of legitimate surface accumulates into a permanent gap in the record.

Squint's approach treats time as part of the problem rather than judging each frame in isolation. Uncertainty on an occasional scene is normal and can be tolerated. Uncertainty that persists in the same region of the input space, orbit after orbit, is a different signal, and it can trigger a change in how the system behaves rather than another individual flag. The distinction is between noise and drift, and it depends on tracking how long a condition has been occurring rather than only what is happening now.

For a satellite, a run of ambiguous calls concentrated on snow-covered terrain would surface as a pattern worth acting on. The response might be to preserve that class of scene more aggressively, or to mark the model as due for retraining on conditions it handles poorly. The gap that nobody on the ground could see becomes something the system can report on its own.

LIMITS AND TRADE-OFFS

Mapping the decision space catches scenes that fall in ambiguous or sparsely supported regions, which is where the costly discards concentrate. It does not guarantee catching a clear image the model confidently reads as ordinary cloud from well inside its trusted region. A map built around where the model lacks support will not necessarily flag a case that looks, to the model, entirely familiar. That is a harder problem, and this approach does not claim to solve it.

Preserving flagged scenes also has a cost. Every image held back consumes some of the bandwidth the filter existed to save, so where the boundary is drawn becomes a mission-level trade-off between downlink capacity and the risk of losing something valuable.

Running any additional module onboard has to fit the spacecraft's compute and power budget, which is a stricter constraint than most embedded deployments face and would need to be established for a given platform.

WHY IT MATTERS

Onboard autonomy earns its place by making decisions no human has time to make. Autonomy without a sense of its own limits leaves the satellite most dangerous exactly where it is least competent. It confidently deletes the rare clear image of a snowfield or a coastline because the surface resembled the clouds it was taught to reject. Nobody can intervene, and no record survives to show it happened.

Giving the system a map of where its own judgment fails lets it keep deciding freely across the vast majority of scenes it genuinely understands, while holding back on the few it does not. For a satellite that cannot ask for help and cannot be second-guessed, knowing which of its own decisions to distrust separates autonomy that saves bandwidth from autonomy that quietly erases what it was sent up to collect.

A representative scenario, grounded in published cloud detection research. Vision Studio's capabilities are described as designed rather than measured from a deployment.